(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

Moving Object Detection
by
Track Analysis

by

Chen-Shan Wang

September, 1990

Thesis Advisor: Chin-Hwa Lee

Unclassified
security classification of this page

REPORT DOCUMENTATION PAGE			
1a Report Security Classification Unclassified		1b Restrictive Markings	
2a Security Classification Authority		3 Distribution/Availability of Report Approved for public release; distribution is unlimited.	
2b Declassification/Downgrading Schedule			
4 Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)	
6a Name of Performing Organization Naval Postgraduate School	6b Office Symbol *(if applicable)* 62	7a Name of Monitoring Organization Naval Postgraduate School	
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	
8a Name of Funding/Sponsoring Organization	8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number	
8c Address *(city, state, and ZIP code)*		10 Source of Funding Numbers	
		Program Element No / Project No / Task No / Work Unit Accession No	
11 Title *(Include security classification)* MOVING OBJECT DETECTION BY TRACK ANALYSIS			
12 Personal Author(s) CHEN-SHAN WANG			
13a Type of Report Master's Thesis	13b Time Covered From To	14 Date of Report *(year, month, day)* SEPTEMBER 1990	15 Page Count 88
16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	HOUGH TRANSFORM, ILS, LAS, SORTING, MODIFICATION, SIMILARITY

19 Abstract *(continue on reverse if necessary and identify by block number)*

The purpose of this research is to study the Hough transform method, as applied to the detection of tracks of underwater moving objects in Lofargrams. The subjects included are the Hough transformation, clustering study, and reconstruction. Two methods, LAS cluster technique and Sorting, are used for cluster analysis. Encouraging results are obtained from the Sorting method. A further improvement of the Sorting is shown to yield better results in processing noisy track data. Experimental results dealing with noise free artificial data, noisy artificial data, and real noisy data are presented. The improved Sorting technique presented in the thesis has shown improvements compared to the straight forward Sorting when it is applied to spectral component tracking.

20 Distribution/Availability of Abstract ☒ unclassified/unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual CHIN-HWA LEE	22b Telephone *(Include Area code)* (408) 655-0242	22c Office Symbol 54Ss

DD FORM 1473,84 MAR | 83 APR edition may be used until exhausted
All other editions are obsolete | security classification of this page

Unclassified

Approved for public release; distribution is unlimited.

Moving Object Detection By Track Analysis

by

CHEN-SHAN WANG
Commander, R.O.C Navy
B.S.E.E, CHUNG CHENG INSTITUTE OF TECHNOLOGY R.O.C. 1977

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
SEPTEMBER 1990

Author: [redacted]
CHEN-SHAN WANG

Approved by: [redacted]
CHIN-HWA LEE, Thesis Advisor

[redacted]
RALPH HIPPENSTIEL, Second Reader

[redacted]
MICHAEL A. MORGAN, Chairman,
Department of Electrical and Computer Engineering

ABSTRACT

The purpose of this research is to study the Hough transform method, as applied to the detection of tracks of underwater moving objects in Lofargrams. The subjects included are the Hough transformation, clustering study, and reconstruction. Two methods, LAS cluster technique and Sorting, are used for cluster analysis. Encouraging results are obtained from the Sorting method. A further improvement of the Sorting is shown to yield better results in processing noisy track data. Experimental results dealing with noise free artificial data, noisy artificial data, and real noisy data are presented. The improved Sorting technique presented in the thesis has shown improvements compared to the straight forward Sorting when it is applied to spectral component tracking.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I am particularily grateful to my advisor, Dr. Chin-Hwa Lee, for his guidance, support and encouragement. I would also like to thank Dr. Ralph Hippenstiel who provided a lot of help and enthusiasm when it was greatly needed. I am most grateful to my wife, Li-Hwa and my parents, for their moral support and encouragement throughout my two years at the Naval Postgraduate School. Finally, thanks go to Mr. Dan Zulaica, Chang-Lung Kao and Chih-Lyeu Chen for their assistance.

I. INTRODUCTION

Track detection and recognition of moving objects in Lofargrams is a subject of interest in image processing. Especially in military application, it is an important problem. The spectral tracks in a lofargram usually exist in a noisy background. To detect and recognize a curve belonging to a target from image data is a problem of importance. This research started with the characterization of the line and curve representation, and then concentrated on studying the Hough transform algorithm.

The main objective of this study is to search for a method that can be used by a computer rather than by a human to detect the tracks of an object and to extract information pertaining to the image data source. The study is aimed at the recognition of the tracks of underwater moving objects with possible dynamic Doppler shifts.

A problem of interest is how to get track information in situations where the object exhibits dynamics which are usually observed as a string of line segments. In addition, how to determine the missing data (i.e. temporary disapperance of the line) from the noisy image data is also an important issue to be studied. These will be discussed as a modification of the Hough transform method.

Before experimenting with the Hough transform method, the image data must be set up in the right format. The image sources can be real data, or artificial data. In the majority of experiments, artificial data is used. For further study, real data can also be used in the computer experiment.

To create artificial data, the user can either write his own program or use the commercial signal processing software package called Interactive Laboratory System (ILS). The tool to display the artificial image data is the IM-4000 Image Manager in the VAX/VMS system. The IM-4000 Image Manager is a software package which can display images up to an array size of 512 × 512 pixels. With 8-bit pixel resolution, it is possible to display simultaneously up to 256 colors or gray shades, which is adequate for the display of most digital images. A functional block diagram of the IM-4000 Image Manager is shown in Figure 1. [Ref. 1]

The Hough transform method allows the detection of line structures in a parameter domain. This method is stable in the presence of noise. Because the cluster finding determines the success of line detection, two methods were used in the study. One is using the routines of the Land Analysis System (LAS), and the other is based on the Sorting technique.

In order to use the routines of the LAS, a splitting technique is used to create two bands of subimages. One is for ρ band and the other is for θ band. After the image splitting, the Land Analysis System (LAS) routines are used to find the position of the clusters. In this study HINDU, KMEANS , ISOCLASS [Ref. 2] routines of the LAS are adopted to do the cluster analysis. Comparisons and results of these routines are discussed in latter chapters.

To confirm the accuracy of the cluster analysis a reconstruction procedure based on the positions of the clusters is obtained from the LAS. If the Sorting method is used, a threshold must be determined. To verify the performance of the Hough transform technique, experiments with noisy images have to be performed. In the reconstruction with noisy images, the Hough transform method can find straight lines easily. For dynamic lines (i.e. curves), there are still problems in the recontruction process which are due to the simplicity of Sorting technique. Hence, we need to improve the Hough method to recognize curves. A later chapter will discuss an extended Hough method that provides satisfactory results. In a real life scenarios the information obtained from the Hough transform method can be used in a strategic decision process for further analysis.

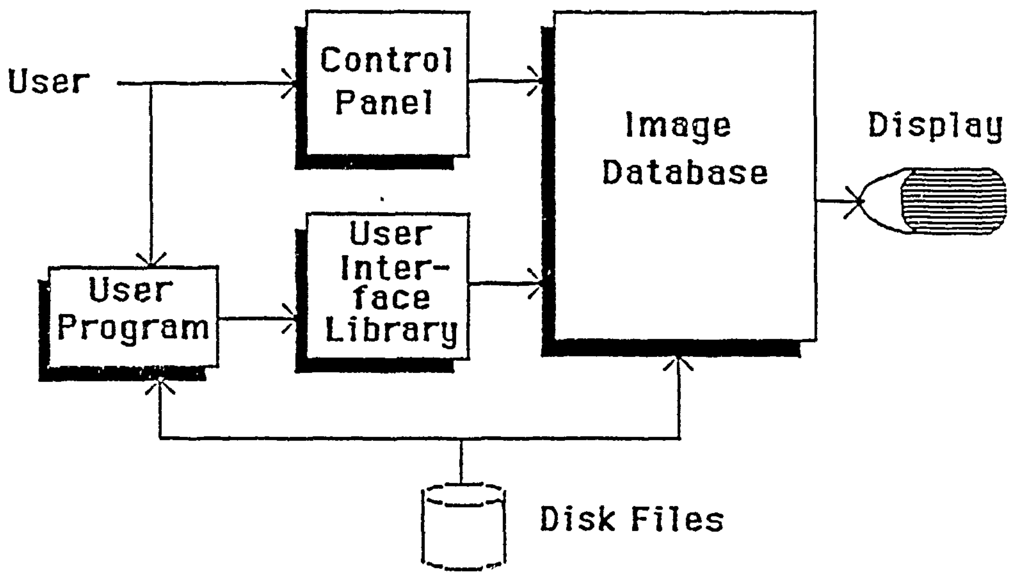


Figure 1. IM-4000 control block diagram (adopted from [Ref. 1]).

There are five chapters in the thesis. Chapter I is a introductory description of the track detection and the line representation problem. Chapter II presents the theory behind the computer simulation. The principle of the Hough transform method and algorithms for curve detection will be discussed. Chapter III emphasizes the detailed implementation of the Hough transform method such as clustering, and reconstruction in noise. Chapter IV shows the results obtained from the computer experiments in the presence of noise. An improvement to obtain satisfactory results is discussed as well as the experiences gained. Chapter V presents the conclusions and recommendations for future study.

II. LINE AND CURVE DETECTION USING THE HOUGH TRANSFORM METHOD

A. TRACK DETECTION PROBLEM

Acoustic track data (i.e images) are usually displayed as intensity variations in a hybrid domain, where the horizontal axis is the frequency and the vertical axis is the time. The received acoustic data is processed to extract the spectral information as a function of time (i.e. waterfall display) and presented in the above image format. A stationary object emitting a constant sinusoidal oscillation will appear as a constant vertical line in the acoustic image. If the object is moving, a time varying Doppler shift is introduced to the signal and the tracks will become curves Under this scenario, the track detection problem becomes a special line detection problem.

B. HOUGH TRANSFORM

1. General Statement

In the field of digital image processing, one of the transform techniques that can be used to detect lines and curves is called the Hough transform. From a mathematical viewpoint, it is clear that the shortest distance between any two separated points is a straight line with the equation expressed as $y = mx + c$, where m is the slope and c is the intercept of the line. In this respect, the line can be assumed to be a set of points with the same value of m and c. The Hough transform method detects lines and curves based on the relationship between a line and the points creating it. In real images, a blurred line can be recognized by the presence of a group of colinear or almost colinear points in a parameter space.

In general, if a picture contains n points (i.e. discrete white points lying on a black background), then there are $\frac{n(n-1)}{2}$ possible lines. It is necessary to perform

$\frac{n(n(n-1))}{2} \sim n^3$ comparisons to determine colinearity. Hence the detection of straight lines is problem in computation and may be prohibitive for large n. In view of the above mentioned difficulty, a method to change the original colinearity problem to one that finds concurrent lines was proposed by Hough [Ref. 3].

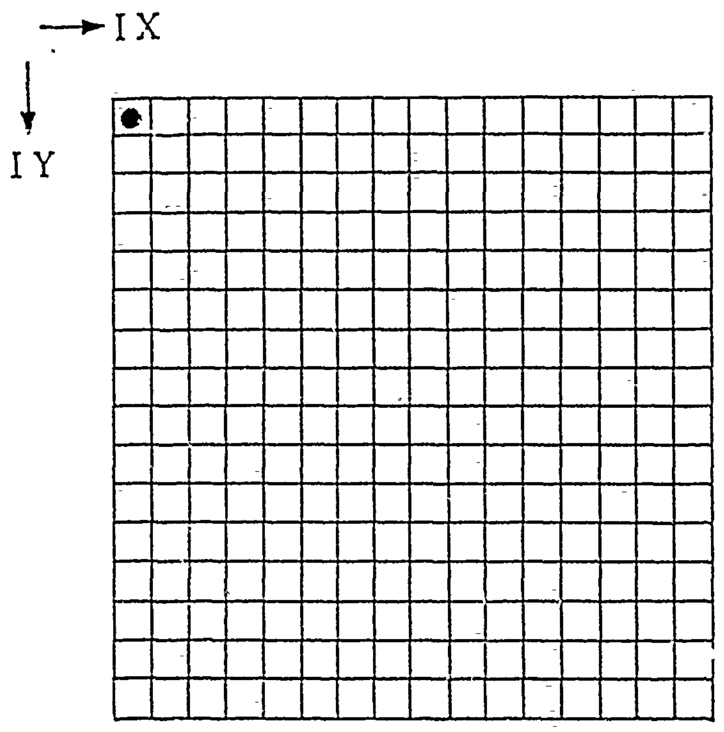
2. Fundamental Theory and Representation

The Hough transform is a mapping between the coordinate of the image space and the coordinate of the parameter space. The transformation equation for a straight line such as y = mx + c in the image space, will create a common point at (m , c) in the parameter space. A problem will occur when the line in the image space is a vertical line since the slope is undefined. An alternative way is to use the (ρ, θ) parameter space instead of (m , c) space. In the (ρ , θ) space a straight line becomes $x\cos\theta + y\sin\theta = \rho$.

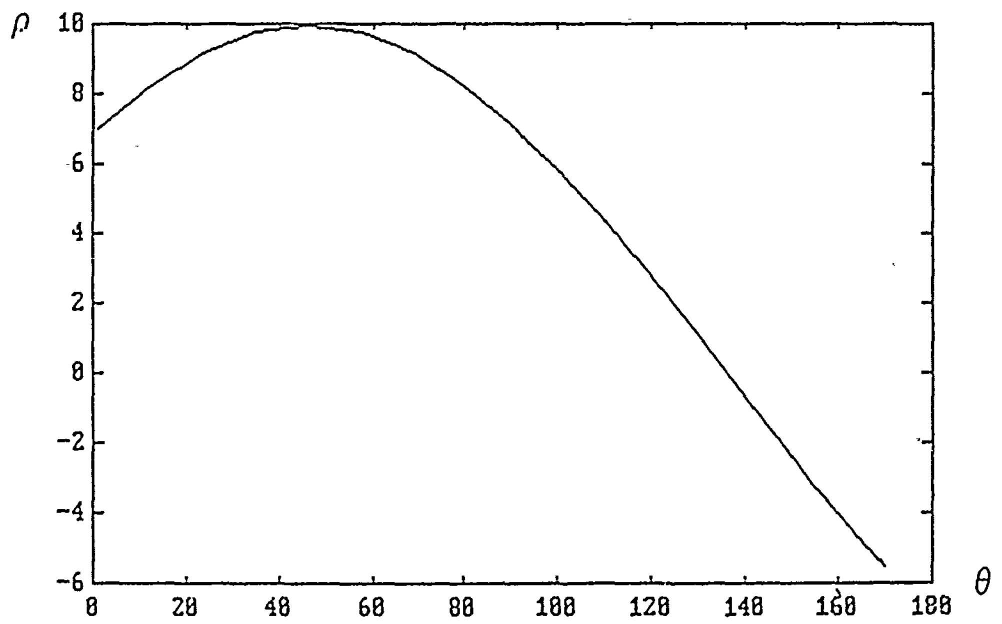
The Hough transform can be used to deal with the detection of specific structural relationships between pixels in an image. In a picture plane, one pixel of the x-y space can correspond to a sinusoidal curve in an (ρ , θ) parameter space according to,

$$\rho = x\cos\theta + y\sin\theta \tag{2.1}$$

Consider the image space in Figure 2(a) with a pixel point at (x_1 , y_1). It corresponds to a curve in the parameter space shown in Figure 2(b). If we add a second point (x_2 , y_2) into the same image space as shown in Figure 3(a), we can get another corresponding curve in Figure 3(b). These two curves intersect at a common point of (ρ , θ). The fact is that any other points lying on the line formed by these two points (x_1 , y_1) and (x_2 , y_2) will correspond to a set of curves, and these curves intersect at the same common point (ρ' , θ') [Ref. 3]. Hence for line detection, as long as we can figure out the location of the common cluster point in the parameter space, we equivalently detected points along a common line in the image space.

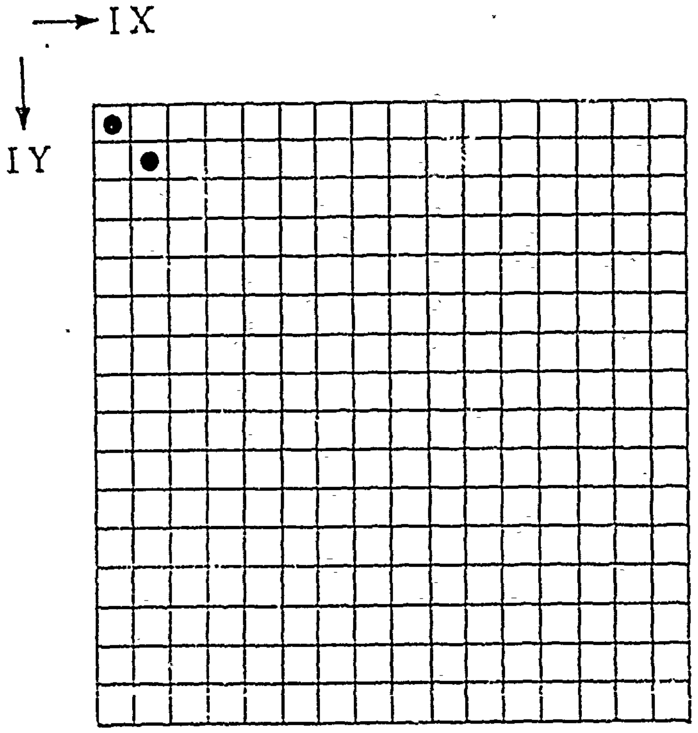


(a) Image space

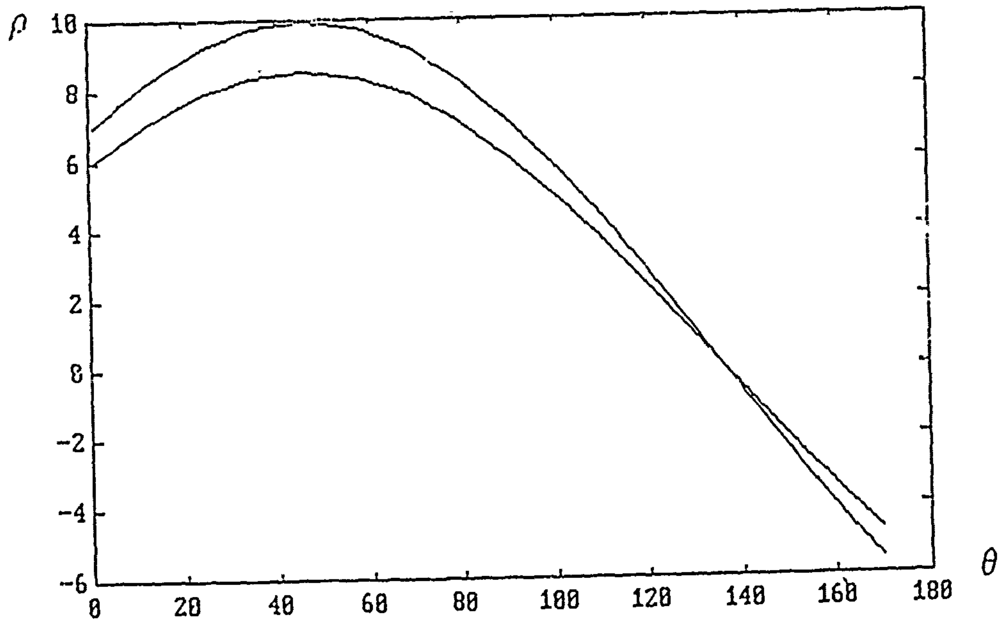


(b) Parameter space

Figure 2. Image space and parameter space transform



(a) Image space



(b) Parameter space

Figure 3. Image space and parameter space transform

There are a number of properties for the Hough transform [Ref. 4]. These properties are:

- A point in the image space (x_0, y_0) correspond to a sinusodial curve $x_0\cos\theta + y_0\sin\theta = \rho$ in the parameter space (ρ , θ).
- A point in the parameter space (ρ , θ) can be used to reconstruct a straight line in the image space.
- Points lying on the same straight line in the image space correspond to sinusodial curves touching at a common point (ρ' , θ') in the parameter space.
- Points lying on the same sinusodial curves in the parameter space correspond to lines through the same point in the image space.

The Hough transform can be used successfully to detect a significant distribution in the parameter space. In the presence of noise, this is complicated by the quantization of both the image (spatial) space axis and the parameter space axis. The pixel value at a spatial position (x , y) is transfered to weight the sinusoidal mapping in the parameter space.

C. CLUSTER ANALYSIS

A cluster is a set of points in the parameter space where the population of points is high compare to the population of parameter points in the surrounding region. Localization of clusters in the parameter space can be used to compress image data. The quantization procedure is an important step in processing the real value of ρ. The smaller the quantized bin size, the more accuracy can be obtained.

In this research, two ways are implemented to find the positions of the clusters. One way uses the software routines of the Land Analysis System (LAS). LAS is an image analysis system designed not only for use with satellite images but also for the purpose of manipulating and analyzing digital image data. It includes a wide range of functions and statistical tools. The second way uses Sorting for cluster analysis.

1. LAS Routines

LAS includes a variety of routines for cluster analysis. Three routines for unsupervised classification (HINDU, KMEANS, and ISOCLASS) are used. They are summarized below.

a. The HINDU Classification Routine

HINDU classifies a multiband im[illegible] based upon its multidimensional histogram which corresponds to the distr[illegible]n [illegible] parametric space in our study. Regions in the histogram with high density are [illegible]arded as clusters. The user specifies the input image, the minimum and maximum acceptable number of clusters, and the histogram bin size.

b. The KMEANS Classification Routine

KMEANS performs an unsupervised classification using the K-means algorithm. The basic K-means algorithm operates as follows.

Step 1: Begin with an arbitrary set of cluster centers for the desired number of clusters.

Step 2: Compute the sample mean of each cluster.

Step 3: Reassign each sample to the cluster with the nearest mean distance.

Step 4: If the classification of all samples has not changed, stop; if not, go to step 2.

c. The ISOCLASS Classification Routine

ISOCLASS performs the unsupervised classification of an image using an ISODATA-type clustering algorithm. The basic ISODATA clustering algorithm operates as follows

Step 1: Set the preset count for printing summary cluster statistics.

Step 2: Cluster the data into C classes. Eliminate any class with fewer than T members.

Step 3: If the preset count has been reached, go to step 6. If $C < 2N$ split any cluster whose samples form sufficiently disjoint groups. If any clusters have been split, go to step 1.

Step 4: Merge any clusters whose means are sufficiently close.

Step 5: Go to step 1.

Step 6: Print the summary cluster statistics, then stop.

In the algorithm description, C is the number of classes, T is the minimum number of distributions allowed in a cluster, and N is the approximate desired number of clusters.

A special splitting technique was adopted when we use LAS to find the clusters in the parameter space. Because LAS accepts band images, it is necessary to split the original image into seperate ρ band images and θ band images. Modification is needed in processing the ρ band because the original ρ band includes negative values which will cause errors in the image display.

2. Sorting Technique

Sorting technique, just as its name implies, is based on rearranging the distribution. Based on the distribution of the maximum cluster, the user can set up a threshold factor to pick up useful clusters that are sufficient large enough in the parametric space. Threshold is inevitably used in both the LAS and the Sorting methods to decrease the noise effect and the processing time. The difference is in the sensitivity of the threshold. Robust threshold is usually preferred. Generally, a clustering algorithm is based on iterative split and merge operations. Consider for example the situation that each cluster picked up from Sorting might include a set of points as shown in Figure 4. These points can result in a worse situation in reconstruction. Hence, it is necessary to use a similarity measure discrimination to merge similar points. By using a similarity measure in Figure 4 a more accurate cluster location can be achieved [Ref. 5: pp. 419]. Figure 5 shows two cluster sites $\bar{v}_i$, and $\bar{v}_j$ given by $\bar{v}_i = (x_i, y_i)$ and $\bar{v}_j = (x_j, y_j)$, respectively. The similarity measure between $\bar{v}_i$ and $\bar{v}_j$ is defined by

$$s(\bar{v}_i, \bar{v}_j) = \frac{<\bar{v}_i, \bar{v}_j>}{<\bar{v}_i, \bar{v}_i> + <\bar{v}_j, \bar{v}_j> - <\bar{v}_i, \bar{v}_j>} \tag{2.2}$$

where

$$<\bar{v}_i, \bar{v}_j> = |\bar{v}_i|\,|\bar{v}_j| \cos(\bar{v}_i, \bar{v}_j)$$
$$= b \times c \times cos\theta$$

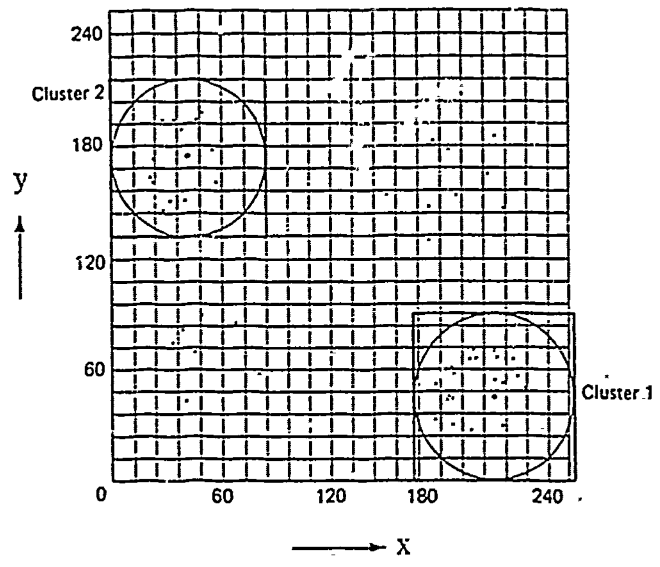


Figure 4. Example of clustering in a parameter space.

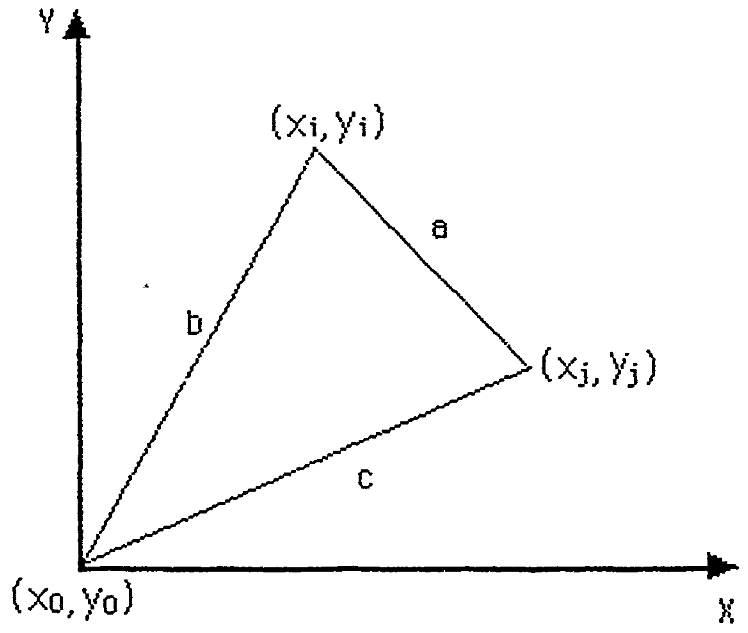


Figure 5. Mathematical relationship of the similarity measure.

and

$$\cos\theta = \frac{b^2+c^2-a^2}{2bc}$$

$$b = \sqrt{(x_i-x_0)^2+(y_i-y_0)^2}$$

$$c = \sqrt{(x_j-x_0)^2+(y_j-y_0)^2}$$

$$a = \sqrt{(x_j-x_i)^2+(y_j-y_i)^2}$$

and (x_0, y_0) is the origin of the parameter vector space, so

$$\begin{aligned} s(\bar{v}_i, \bar{v}_j) &= \frac{<\bar{v}_i, \bar{v}_j>}{<\bar{v}_i, \bar{v}_i>+<\bar{v}_j, \bar{v}_j>-<\bar{v}_i, \bar{v}_j>} \\ &= \frac{b\times c\times\cos\theta}{b^2\times c^2-b\times c\times\cos\theta} \qquad (2.3) \\ &= \frac{b^2+c^2-a^2}{b^2+c^2+a^2} \end{aligned}$$

The similarity measure is tested against a threshold. If the similarity measure is less than the threshold, v_i and v_j sites will be merged.

D. RECONSTRUCTION

From the result of running the LAS routines or the Sorting program, the positions of the clusters are known. The value at the position of the parameter of ρ and θ is the accumulation due to the transform and the track's intensity. Hence the reconstruction can follow a reverse procedure as follows.

For a given image, if we can figure out the cluster in the parametric space, then there is no problem to reconstruct the cluster center (ρ , θ) by following the equation

$$x = \rho - y\frac{sin\theta}{cos\theta} \tag{2.4}$$

In this study, we will select a reference point which is located at the center of the image space by shifting an amount of x_0 and y_0 , the new equation is

$$\rho = (x - x_0)cos\theta + (y - y_0)sin\theta \tag{2.5}$$

and the new reconstruction equation become

$$x = x_0 + \frac{\rho - (y - y_0)sin\theta}{cos\theta} \tag{2.6}$$

E. TRANSFORM METHOD

From the above analysis, a computer experiment can be performed step by step in accordance with the flow diagram in Figure 6.

The procedure consists of four parts. The first part is required to generate an image and noise, to form an image with the desired signal to noise ratio (SNR). A program for noisy image source generation is listed in Appendix A. Also an image with a given SNR can be created using ILS. The operational procedure is descriped in Appendix B. The second part is the Hough transform method where distribution in ρ, θ parameter space are generated and quantized. The third part uses one of two different methods to find the positions of clusters. The fourth part is to reconstruct for experiment verification the image from the selected cluster position information.

The relation between image space and parameter space, as described previously, suggests the following algorithm for line detection [Ref. 6].

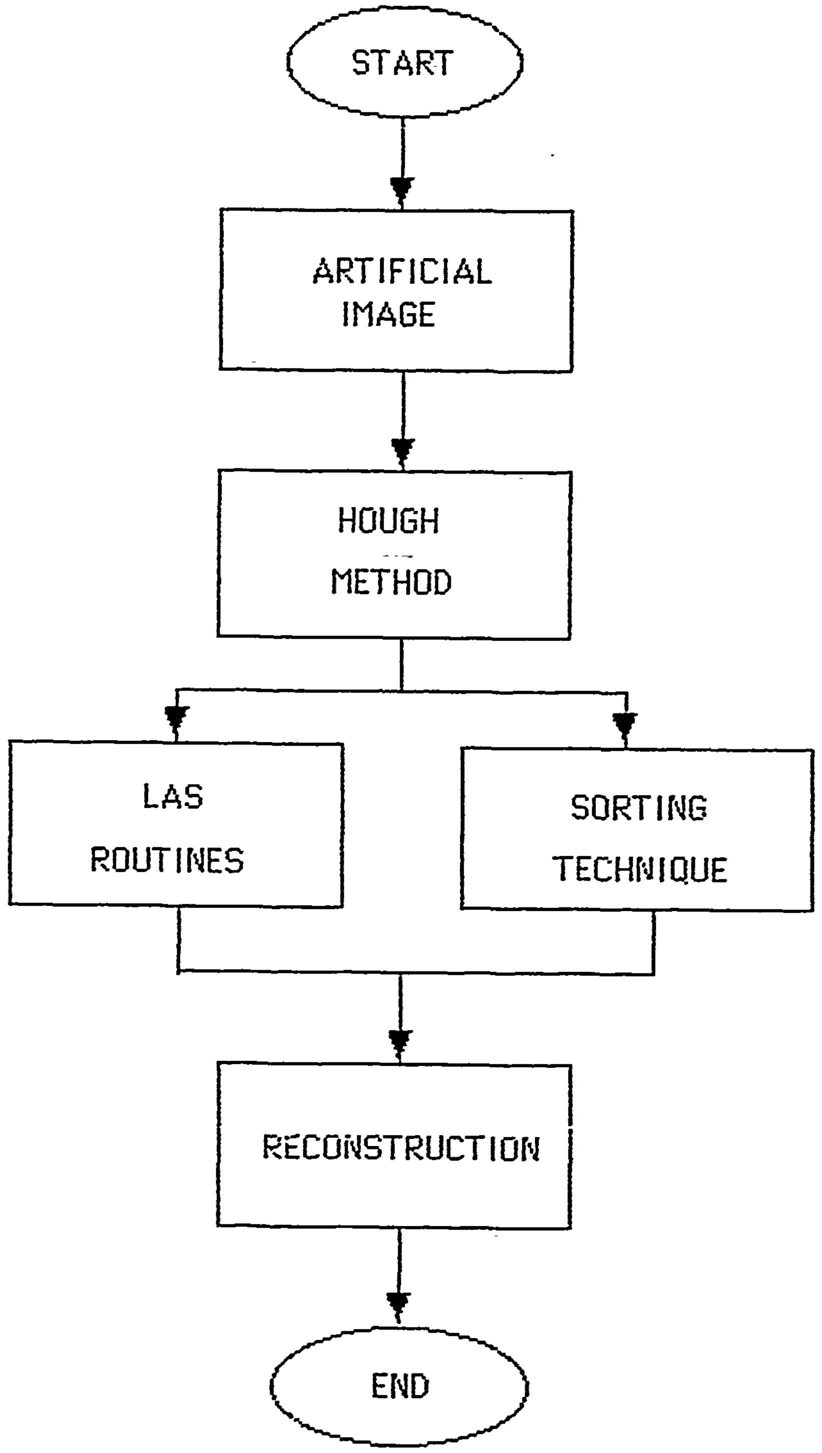


Figure 6. The flow diagram of the computer simulation.

- Table look up technique is used for sinθ and cosθ calculation, which are required in the computations in the parameter space.
- Select the center of the image space as the reference point.
- Compute the corresponding ρ value by using equation (2.5) and store these values in a two dimensional array.
- Quantize the ρ values and θ values.
- Create an accumulator array jrt(θ , ρ) , whose elements are initially set to zero.
- Increment all quantized points in the accumulator array along the appropriate line as follows

$$jrt(\theta, \rho) = jrt(\theta, \rho) + ipd \qquad . \quad (2.7)$$

$$ic = ic + 1$$

  for θ and ρ satisfying equation(2.5). Here ic is used as a counter, ipd is the gray level of the pixel that is added to each element in the accumulator array.

- Any value that is greater than the threshold located in the accumulator array will correspond to the most likely set of colinear points in the image space.

F. NOISY IMAGE DATA

In the ideal case, good result can be achieved if there is no noise involved. But, it is inevitable to encounter a blurred image in a noisy environment . Hence , how to decrease or eliminate the effect of noise to recognize the track is an important problem to be studied.

In this study, artificial image data is generated to avoid the use of classified information as well as to have truth for comparion available. The noise is added into a noiseless picture to provide a desired SNR [Ref. 7],

$$SNR(dB) = 10 \log \frac{(RMSS)^2}{c \times (RMSN)^2} \tag{2.8}$$

where RMSS is the root mean square value of signal, RMSN is the root mean square value of the noise. For a given SNR, there exists a proportional constant relationship between SNR and c. Hence when a user decides to create a SNR(dB) image, the constant c can be computed from equation (2.8). Accounting for c in the required noise variance to obtain the desired artificial image, is straight forward.

The principle of generating a desired SNR image should be based on the spectral domain. That is, when adding the noise to a pure signal, it makes sense to add signal in the spectral domain, which is equivalent to add noise to the signal in the time domain. The additive noise is Uniform noise.

The generated image should be a normalized image to be displayed using the IM-4000 software package for verification in the process. Hence, for any input image it is necessary to test the gray level for each pixel. The value of the gray level should be restricted to an integers between -128 and 127 as a signed number or 0 to 255 as an unsigned number for the IM-4000 software.

III. IMPLEMENTATION DETAILS OF THE COMPUTER SIMULATION

In this chapter, more details about the implementation of noise and signal generation, Hough transform method, and reconstruction are described.

A. SIGNAL AND NOISE GENERATION

According to ILS tools noise is created randomly. The image can be generated in accordance with the steps in appendix B, where the required SNR can be obtained by adjusting the proportional constant c.

$$c = \frac{RMSS}{RMSN \frac{e^{nxln10}}{20}} \qquad (3.1)$$

B. HOUGH TRANSFORM METHOD

The Hough transform method is applied only to pixels having values other than 0 in the image array which requires the output from the IM-4000 package to be of the unsigned form. The reference point used for this method is the origin of the coordinates, as shown in Figure 7.

Based on Figure 7, the range of ρ becomes

$$-r_{max} \leq \rho \leq r_{\max} \qquad (3.2)$$

where

$$r_{\max} = \frac{\sqrt{ix_{\max}^2 + iy_{\max}^2}}{2}$$

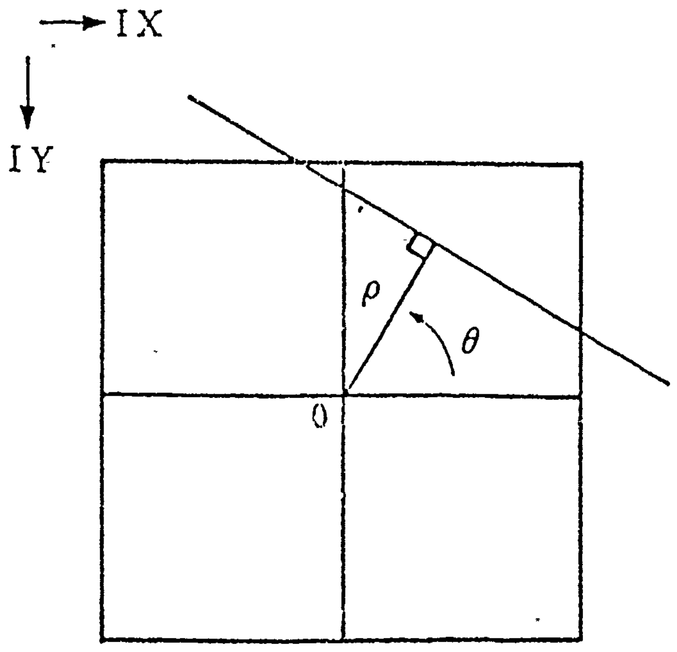


Figure 7. The reference point in image spatial domain.

In the parameter space, the ρ axis is quantized into 256 bins while θ is quantized into 180 bins. However, any value of ρ calculated from equation (2.5) is a real number which will have to be quantized into an integer. After obtaining the integer, the negative ρ values still need to be mapped into positive ρ values to avoid an image display error. Besides, the positive ρ value is necessary for the corresponding quantized accumulative array to have an accumulative distribution. In order to meet the above requirements, the processing is done in the following fashion

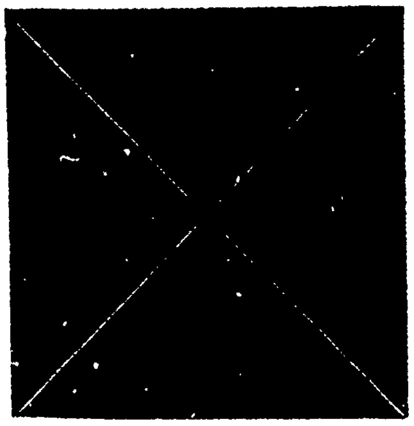
$$IR = \frac{\rho + r_{\max}}{\sqrt{2}} \tag{3.3}$$

where IR not only stands for the positive ρ value but also stands for the corresponding position of the ρ axis in the parametric space. For example, take an artificial image in Figure 8(a), after executing the equation (3.3) every ρ value transfered from the image space will be accounted for in the accumulative array to form an image array. IM-4000 can display this image array as shown in Figure 8(b). The more often a given position of the ρ in the parameter space occurs, the bigger the value of the accumulative array and the brighter the gray level as shown in Fig 8(b). Any value of the element of the accumulative array greater than the selected threshold will represent a possible cluster. For illustration, the 3-D plot of Figure 9 shows peaks which represent the possible clusters.

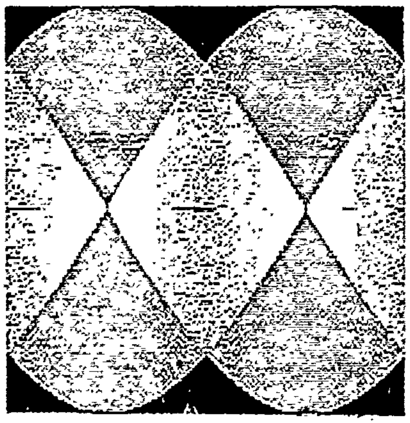
The next step is to find out the positions of the clusters. Two ways are used. One is to use the LAS routine, and the other is to use Sorting.

1. Running LAS Routines

HINDU, KMEANS and ISOCLASS are three different routines that are used to find the sites of the clusters. Because they require a multiple band image as input, it is necessary to provide a normalized ρ band image and a θ band image. Two 2-D arrays



(a) An artificial image.



(b) Result after Hough transform.

Figure 8. Image before and after Hough processing.

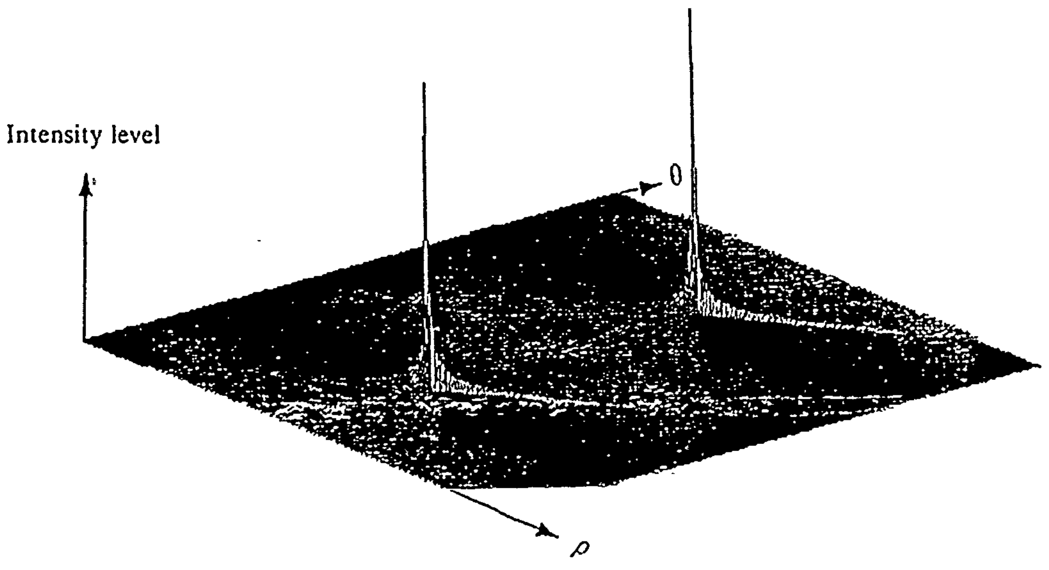


Figure 9. 3-D plot of the (ρ , θ) parametric space.

are used to store the value of ρ as well as the the value of θ. The ρ band image stores the mapped ρ values which are originally obtained from equation (2.5). Before the ρ band image is displayed, the negative ρ values must be mapped into positive ρ values to avoid an error both in future cluster localization and in the image band display. The θ band image stores the θ values which are sequentially assigned between 0° and 180° for each row. For the example of Figure 8(a), we have split it into the two bands as shown in Figure 10(a) and Figure 10(b). While running the LAS routine the two band image data will contribute to the cluster.

2. Normalization

Normalization is a necessary step to transfer the negative value of ρ into a positive value required to execute LAS. The computation follows the formula

$$\rho' = (\rho - \rho_{\min}) \times factor1 \tag{3.4}$$

where

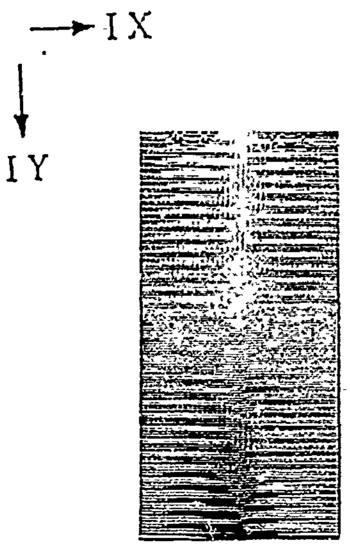
$$factor1 = \frac{255}{\rho_{max} - \rho_{\min}}$$

$$\theta' = (\theta - \theta_{\min}) \times factor2 \tag{3.5}$$

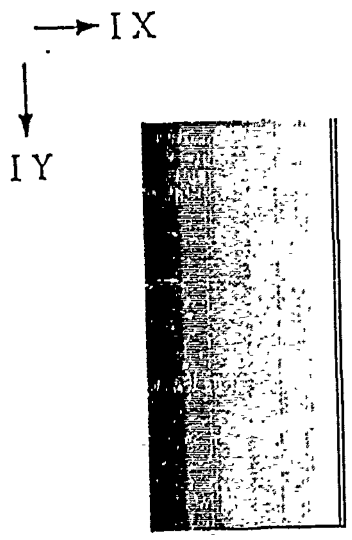
where

$$factor2 = \frac{255}{\theta_{max} - \theta_{\min}}$$

The LAS routine will accept the two band images, then use merge and splitting techniques to eliminate the undesired clusters. The result is statistical information about the positions of the clusters.



(a) ρ band image.



(b) θ band image.

Figure 10. **Two band images of Figure 8.**

3. Sorting Implementation

There is a direct way to find the sites of the maximum magnitude in the parameter space. To determine the clusters a threshold is necessary in this approach. In the Sorting procedure, the value that is stored in the first address of the jrt array will be assumed to be the maximum value. It is compared with the value stored in the next address to find a bigger value to replace the previous maximum value. Repeating the comparison throughout the whole array, the peak value of the magnitude in the parameter space can be obtained (i.e. bubble sort). An experimental threshold is selected by the user such that the number of peaks corresponds to the number of lines. Figure 11 shows the Sorting procedure diagram.

C. RECONSTRUCTION

The value of the positions of the clusters obtained from the result of LAS are not the final information. They need to be transfered back by using the formula

$$\rho = \frac{\rho'}{factor1} + \rho_{\min} \qquad (3.6)$$

$$\theta = \frac{\theta'}{factor2} + \theta_{\min} \qquad (3.7)$$

If we apply these values to equation (2.5) and equation (2.6), the lines can be reconstructed. For a convenient comparison, the example in Figure 8(a) is shown in Figure 12(a) again, and the result of the reconstruction is shown in Figure 12(b).

In this chapter miscellaneous details of the procedures of the subroutines have been discussed. It provides an explanation to readers wanting to execute the program. The next chapter will be aimed at discussing the experimental results.

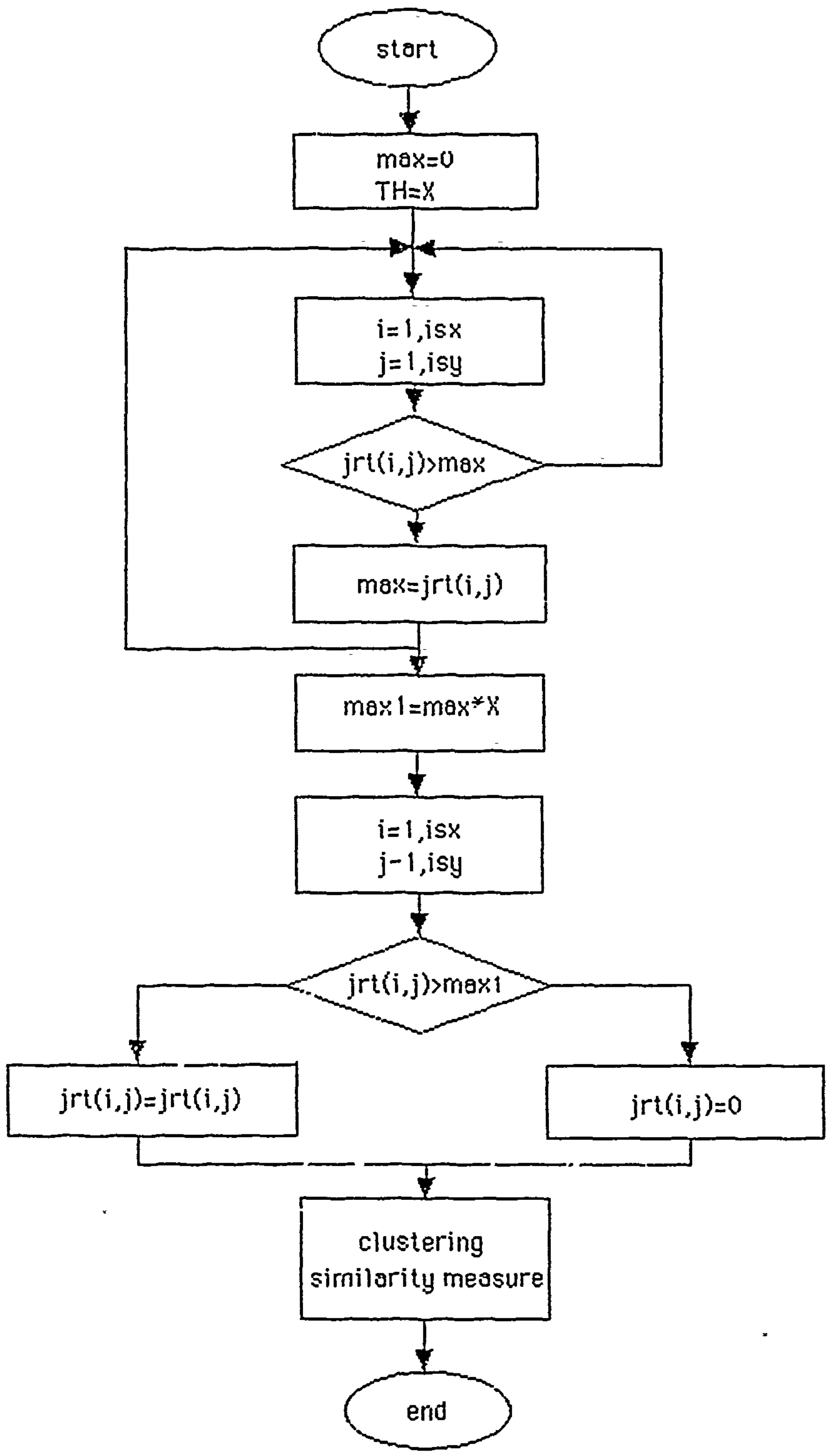
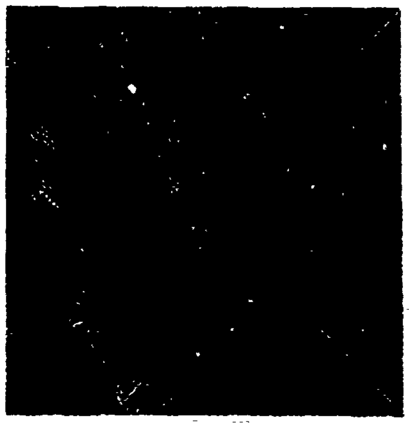
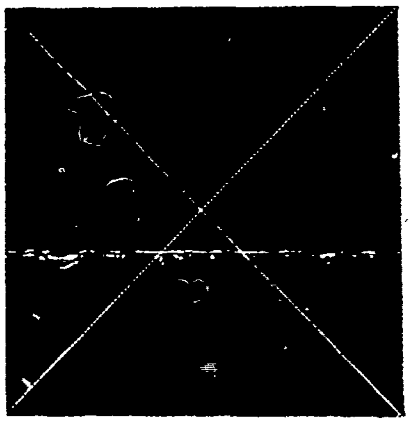


Figure 11. Sorting procedures diagram.



(a) Image of Figure 8(a).



(b) Reconstruction.

Figure 12. Image example reconstruction.

IV. EXPERIMENTAL RESULT

In this chapter the experimental results and performances of different approaches .re presented. These experimental results reconfirm the potential of the Hough transform method. They also show that the noise has less effect on the performance when compared to other image detection systems such as the Affine Invariant Matching algorithm [Ref. 8]. During the experimental procedures, the cluster analysis is done using one of two different ways, the LAS routines or Sorting. In using the LAS routines, there are problems that can not be solved easily. This will be explained at the end of this chapter. Since the study is concentrated on the track analysis of the moving object, the image data is usually a segment of a line or a curve. The tracks of a moving object were generated artificially in the computer. There are three steps in the execution of the experiments. First we work with the noise free environments. Then we work with the noisy signal to test the tolerance of the procedures at various SNR's. And finally we look at ways to improve the weakness of the Sorting method to obtain more reliable results.

A. EXPERIMENTAL SORTING RESULTS

1. Noise Free Tests

First, an ideal noise free situation is considered. Artificial images with three types of tracks are used. These types are lines, curves, and mixed curves.

a. Line Test

A general line test is shown in chapter II as an example. In this section, a special case is considered. We assume that the unknown artificial image has two vertical parallel lines, see Figure 13(a). According to the Hough transform two straight lines will result into two clusters. But the results show four clusters as seen in Figure 13(b). In terms of the 3-D plot, the corresponding clusters are shown in Figure 13(c).

The reason for causing four clusters is that the cluster points will be located at both the $\theta = 0^\circ$ and the $\theta = 180^\circ$, which is due to the trigonometric character of the transformation. Two cluster points exist for each line. One is at $\theta = 0^\circ$, and the other occurs at $\theta = 179^\circ$. In reality there are only two clusters or two pairs of points. During reconstruction each pair should reconstruct its original line. The reconstruction is shown in Figure 13(d) proves that the cluster pair explanation is correct.

b. Curves Test

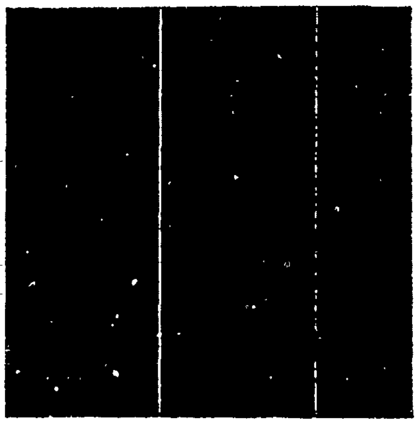
In this section, the assumed curve image has a shape of the letter S as shown in Figure 14(a). By using the Hough transform the clusters are not easily found from Figure 14(b) or Figure 14(c). The result of the reconstruction is shown in Figure 14(d). Though the Figure 14(d) is not the same as the original shape, the intersecting lines more or less provide some information about the moving track. The frequency shift and hence differential speed of a moving object can be estimated from the data in Figure 14(d).

c. Mixed Test

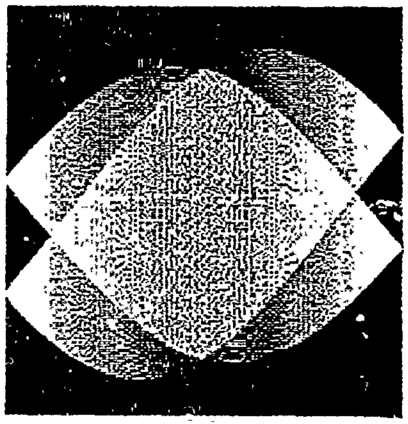
A S-curve and a vertical line represent the mixed test and are shown in Figure 15(a). The Hough transform yields Figure 15(b) and the 3-D plot in Figure 15(c). The reconstructed image is given in Figure 15(d). The reconstruction is not identical to the original so that some modification should be made.

2. Noisy Test

The noisy images are generated by using the ILS software. In this type of experiments, signal to noise ratios are varied to test the tolerance of the procedures. The tests include four different line images with SNR's of 20dB, 8dB, 3dB and -3dB.

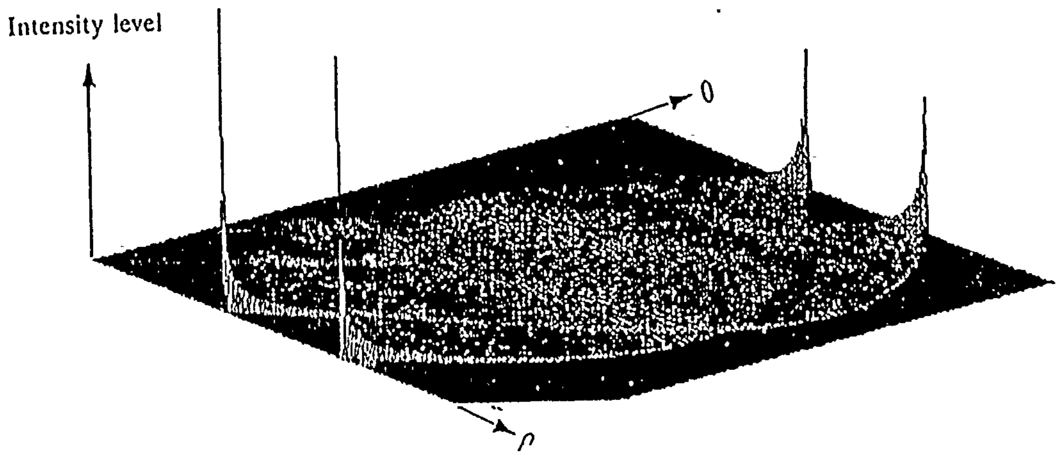


(a) Artificial image(straight line).

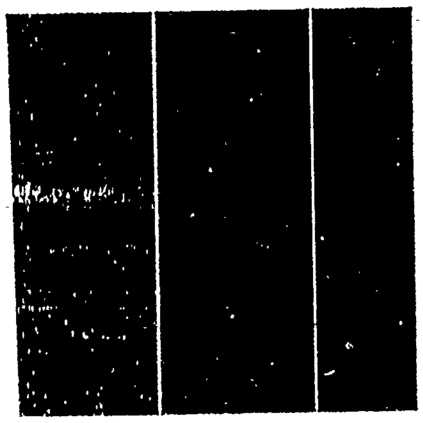


(b) The dense region indicate the clusters.

Figure 13. Line detection in a noise free situation.

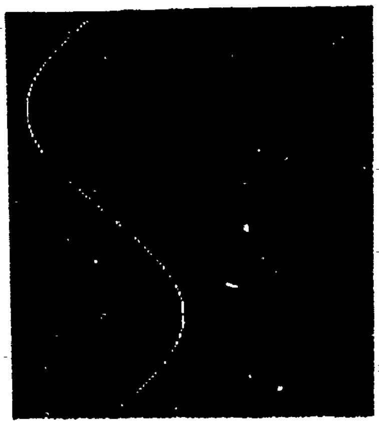


(c) 3-D plot of possible clusters.

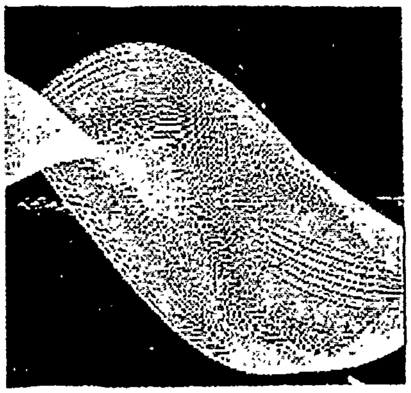


(d) The result of reconstruction.

Figure 13 (continued).

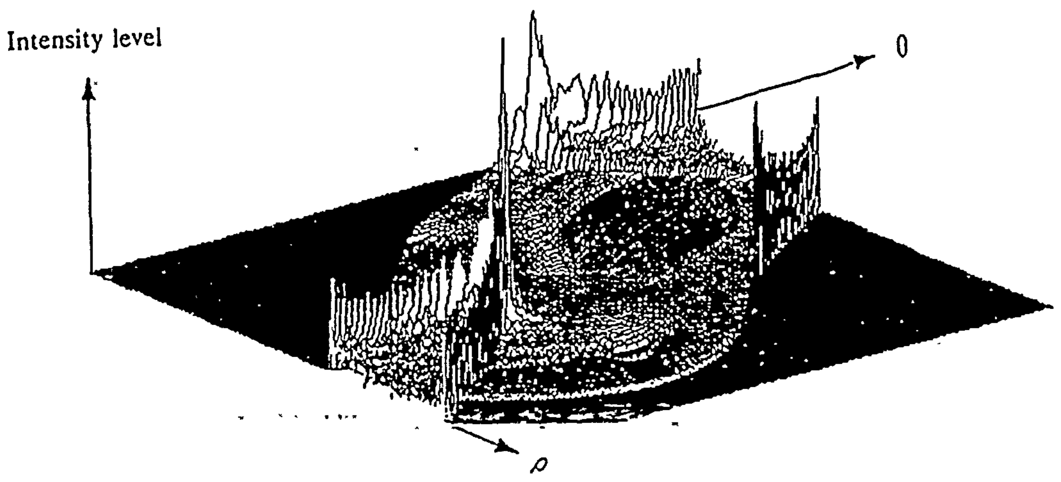


(a) Artificial image(s-curve).

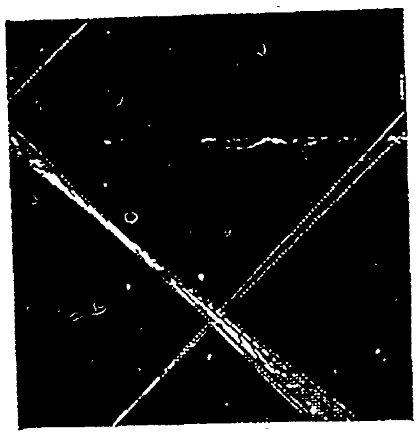


(b) The dense region indicate the clusters.

Figure 14. Line detection in a noise free situation.

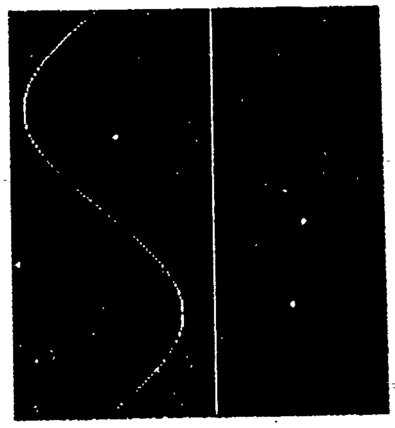


(c) 3-D plot of possible clusters.

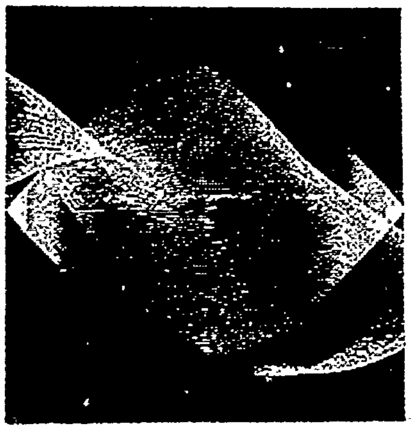


(d) The result of reconstruction.

Figure 14 (continued).

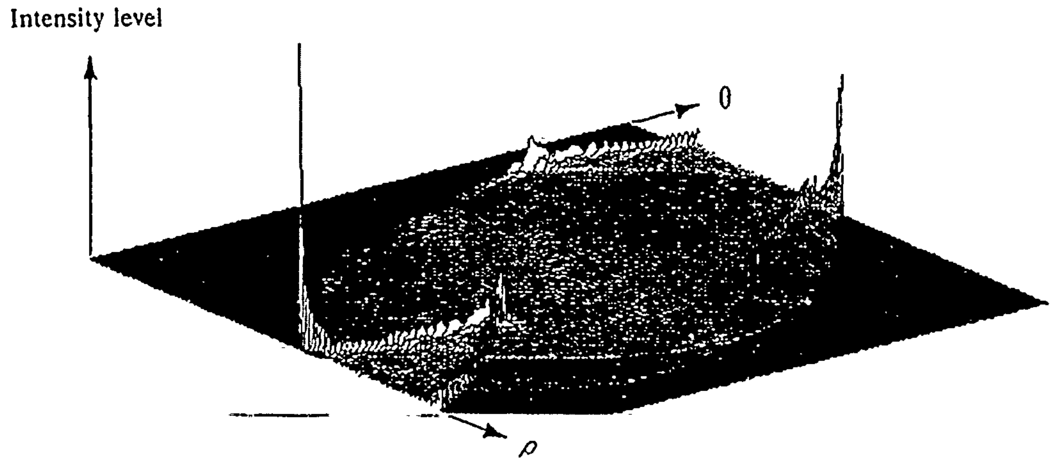


(a) Artificial image (s-curve and line).

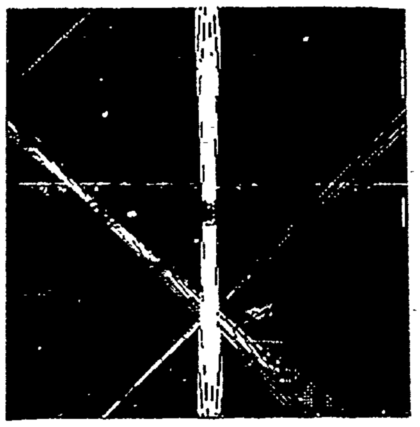


(b) The dense region indicate the clusters.

Figure 15. Mixed line detection in a noise free situation.



(c) 3-D plot of possible clusters.



(d) The result of reconstruction.

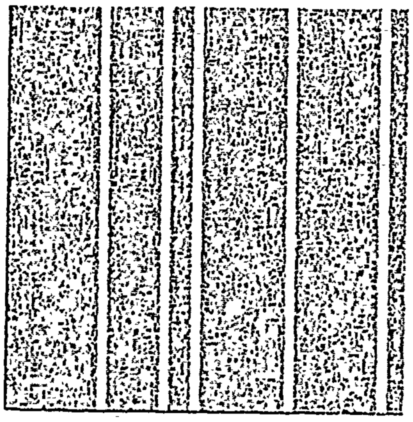
Figure 15 (continued)

a. Line Test

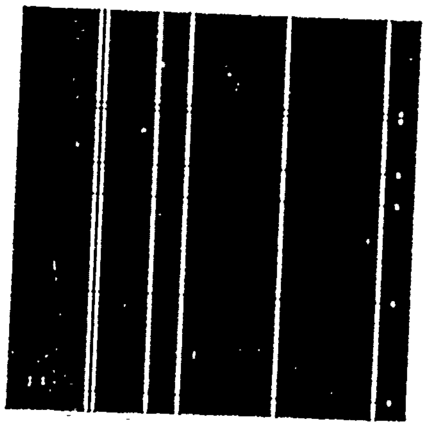
The image consists of five vertical lines formed in ILS with five frequencies of 30, 50, 60, 90, 120 Hz respectively. In this experiment the tracks are identified in the scene despite the noise. Though, a filter can be applied to remove most of the noise, it is time consuming in running filtering while processing long data sequences. The Hough transform can solve the problems, but there is a need to determine the threshold to pick up a sufficient amount of clusters. Figure 16(a) shows the artificial image with a SNR of 20dB. Using the factor of 0.7 as the experimental test threshold, the test reconstruction recover the tracks as can be seen in Figure 16(b). Note the line corresponding to the lowest frequency has been split in the reconstruction. At this time no explanation for this phenomenon has been found. Also, a modification of the algorithm eliminates this potential problem. Repeating the test by changing the SNR to 8dB is shown in Figure 17(a). With the same threshold, the reconstructed results are shown in Figure 17(b). There is still good matching between the signal and the result except for an artifact of a swept line. The test image for 3dB and -3dB are shown in Figure 18(a) and 18(b). The reconstructed images are shown in figure 18(c) and figure 18(d). The swept line which shows up in Figure 18(c) and 18(d) is due to noise since in a noise free simulation no swept line is found. These results can be improved by modifing the Hough transform as discussed in the later part of this chapter.

b. Curve Test

The S-curve with a 3 dB SNR is shown in Figure 19(a). The reconstruction is shown in Figure 19(b). It is obvious that the reconstructed image can not provide the desired information. Hence, for curve detection it is necessary to make a modification in the Sorting procedure discussed in the next section.

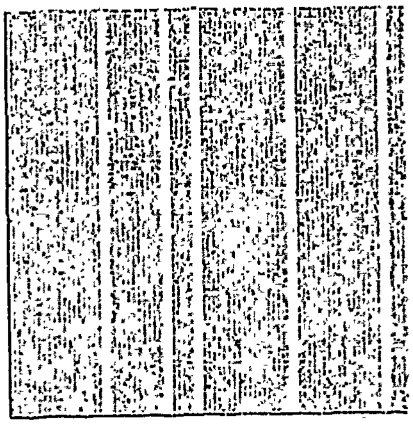


(a) Artificial image (line with SNR 20dB).

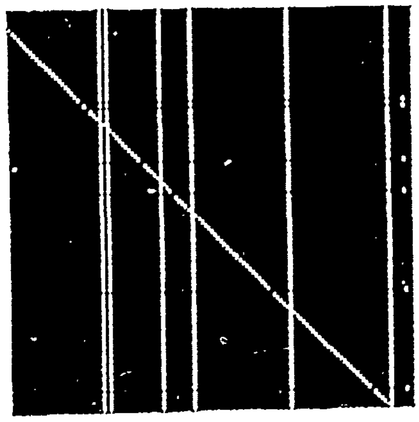


(b) Reconstruction.

Figure 16. Line detection with SNR of 20dB.

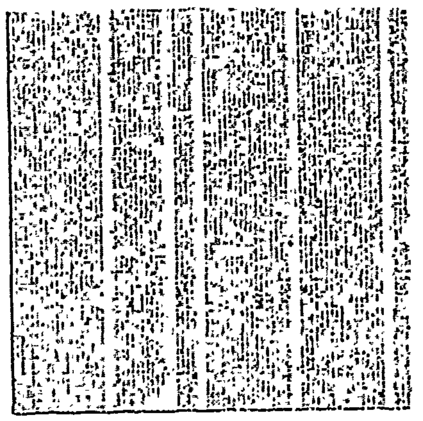


(a) Artificial image (line with SNR 8dB).

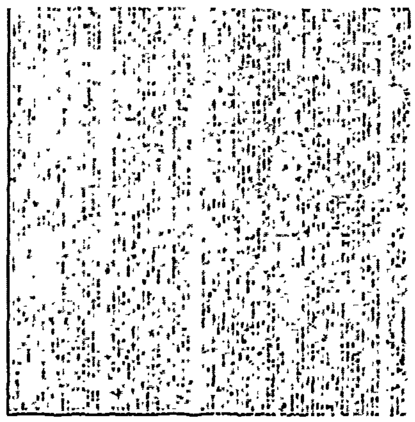


(b) Reconstruction.

Figure 17. Line detection with SNR of 8dB.

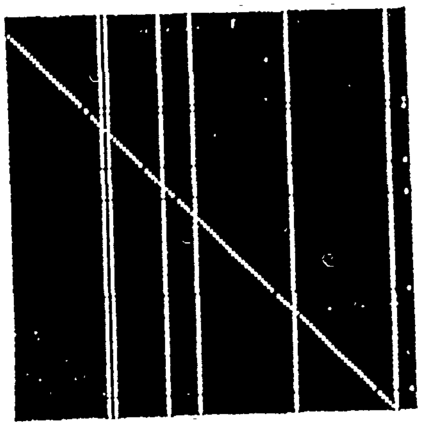


(a) Lines with SNR of 3dB.

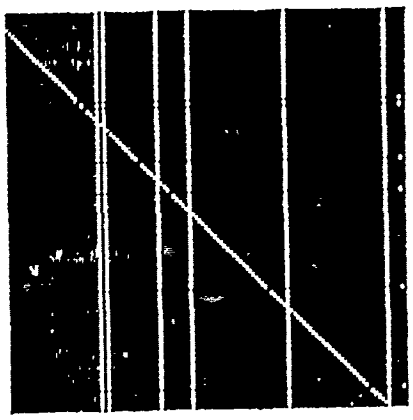


(b) Lines with SNR of -3dB.

Figure 18. Lines detection with SNR (a) 3dB (b) -3dB.

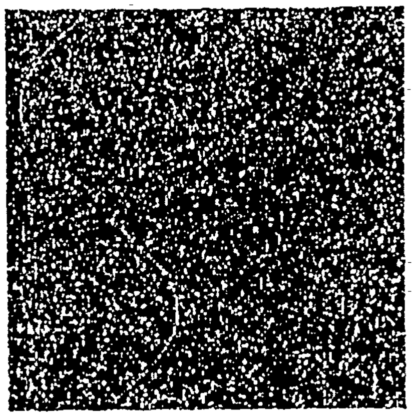


(c) Reconstruction of 3dB image.

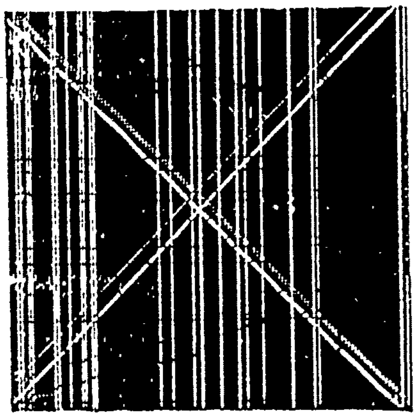


(d) Reconstruction of -3dB image.

Figure 18 (continued)



(a) Artificial image (curve in noise).



(b) Reconstruction (original approach).

Figure 19. Curve detection in noise.

c. *Mixed Test*

This time, a Lofargram is taken as an input as shown in Figure 20(a). Here the signal to noise ratio is unknown. The Figure 20(b) shows the result of the Hough transform. Again, the result is very poor. From the above experiments we see that the noise tolerance of the line detection procedure is good, while the noise tolerance for curve detection is poor.

B. IMPROVEMENTS

1. Algorithm Modification

The Hough transform method has difficulties in processing curves since it is difficult to pick up the real significant clusters without picking up the clusters that are caused by noise. Hence, the first thing to overcome is the influence of noise. One simple step is to split the whole image into smaller segments, then apply the Hough transform to each segment. Finally we put each result to its corresponding place in the splitted image. The modified algorithm is presented as following

Step 1: Split the whole image into N segment.

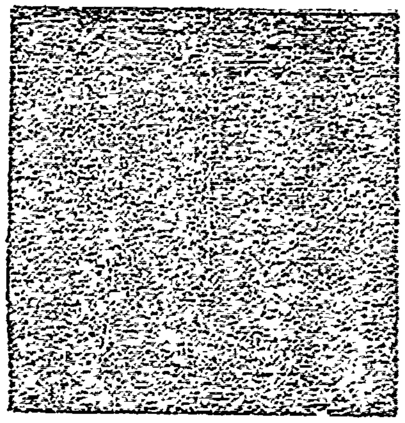
Step 2: Start from the first segment and compare A to B, where A is the mean value of the segment, B is the mean value of the whole image.

Step 3: If A > B, then, run the Hough transform to the segment, else skip to analyze the next segment.

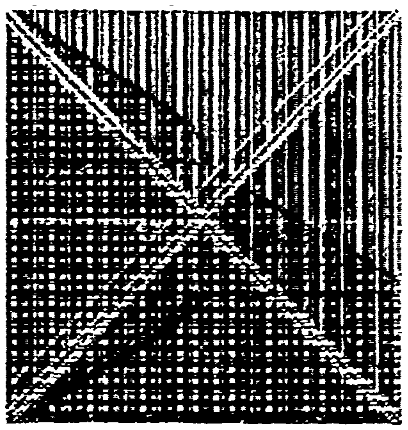
Step 4: Continue step 3. Stop when the last segment had been compared.

2. Results

The Figure 21(b) shows the improvment. In principle, we have eliminated the swept line caused by the noise in Figure 21(a). Figure 22(b) shows the processed result of the artificial image of Figure 22(a).
Figure 22(c) shows the image of the clusters. Figure 22(d) shows the reconstruction from (c). Figure 23(b) shows the processed result of the artificial image from Figure

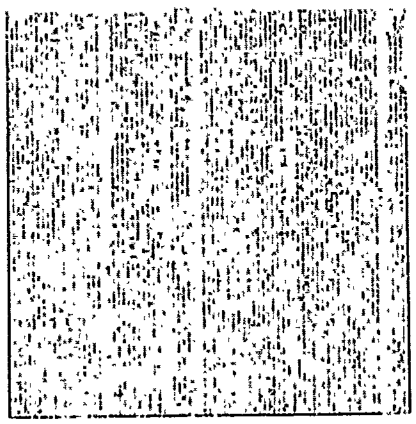


(a) Artificial image (mixed in noise).

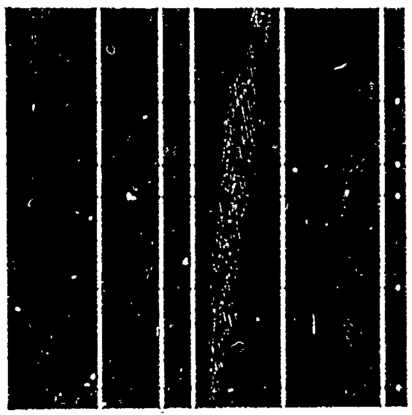


(b) Reconstruction.

Figure 20. Mixed curve detection in noise.



(a) Artificial image (same as Figure 18(b)).



(b) Reconstruction (after modification).

Figure 21. Lines detection after modification from Figure 18(b)

23(a). Figure 23(c) shows the Hough transform results. Figure 23(d) gives the reconstruction from (c).

3. Application of the Result

The improvement mentioned above shows better tracking of moving object, but multiple lines are still reconstructed. What needs to be done, is to analyze the reconstruction. Take the Lofargram shown in Figure 23(a) as an example, it is an image of the time-frequency space. The reconstruction from the image clusters in Figure 23(c) will generate the corresponding lines. In a noisy environment, the set of the reconstructed lines, shown in Figure 23(d), is not good enough for further analysis. Here, a repetition of the Hough transform technique is suggested. Take the image in Figure 23(d) as an input image to a second Hough transform to obtain the clusters image shown in Figure 23(e). By using a clustering method, the clusters can be reduced as in figure 23(f). Once the position of the reduced clusters obtained, we can implement the reconstruction and get a satisfactory final result as shown in Figure 24. From this procedure slope and breakpoints of the track can be obtained for further analysis.

C. EXPERIENCE GAINED

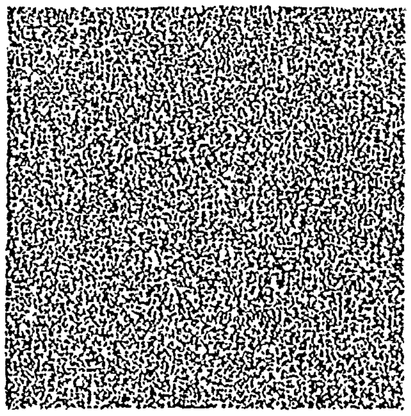
There are several points learned from the experiments:

1. Unique Coordinate Reference

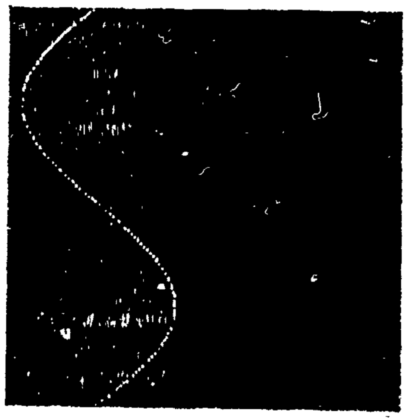
The coordinate transformation between the image space and the parameter space is important not only before cluster analysis for accuracy, but also after cluster analysis for reconstruction. Hence, it is necessary to define a set of unique coordinates to avoid errors. For example, the i and j used in the image space corresponds to θ and ρ in the parameter space, and will be used throughout the program.

2. Boundary Rule

The range of θ is defined as $0^\circ \leq \theta < 180^\circ$ by Hough. The reason to check according to this definition is to avoid generating cluster pairs in an ambiguious fashion. This was discussed in connection with the cluster ambiguity in chapter III.

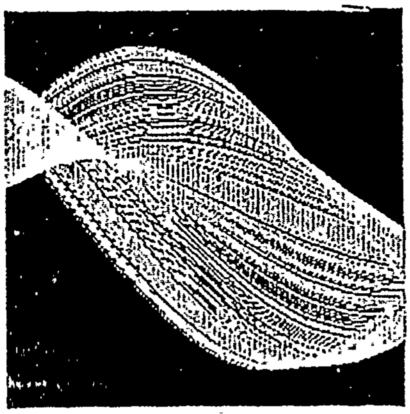


(a) Artificial image (s-curve in noise).

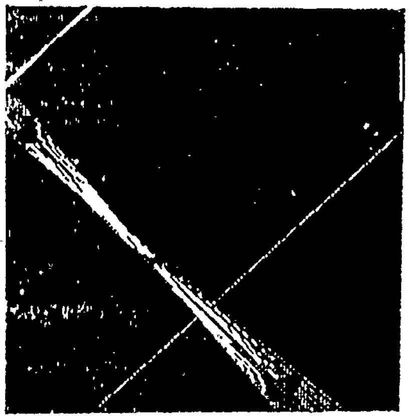


(b) Reprocessed artificial image.

Figure 22. Curve detection in noise.

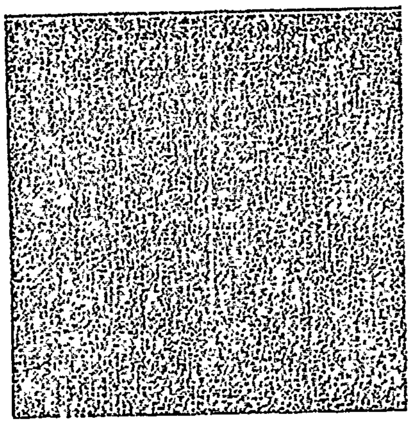


(c) The image of the clusters.

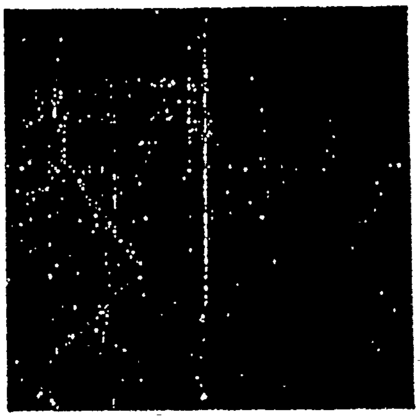


(d) Reconstruction from (c).

Figure 22 (continued)

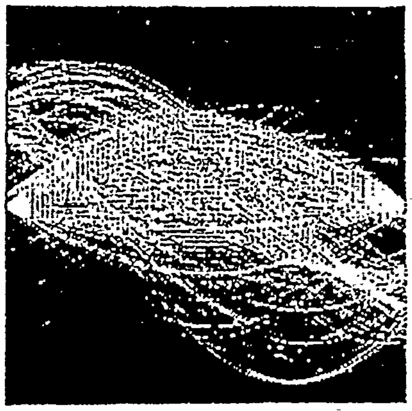


(a) Lofargram image.

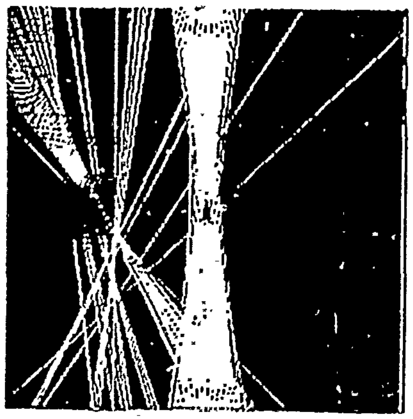


(b) The reconstructed Lofargram.

Figure 23. Curve and line detection on a Lofargram.

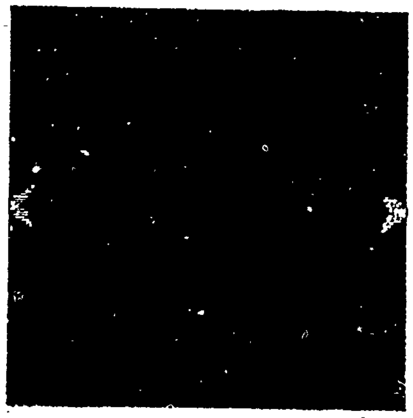


(c) The image of the clusters of the Lofargram.



(d) Reconstruction from (c).

Figure 23 (continued)



(e) The clusters from (d) by a second Hough transform.

(f) The reduced clusters obtained from (e)
Figure 23 (continued)

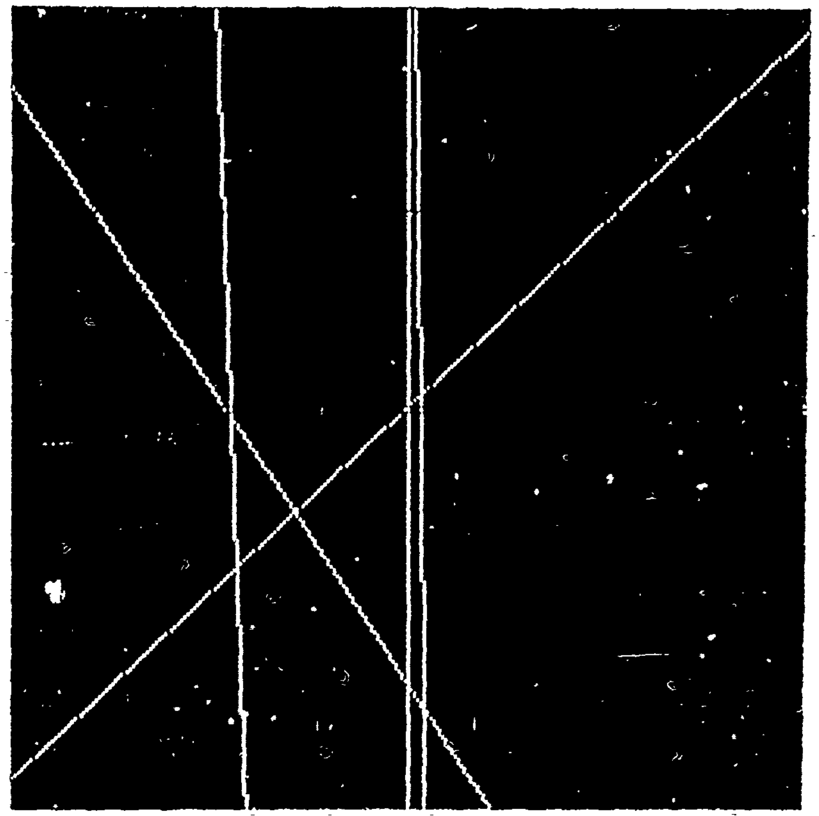


Figure 24. Curve and line detection on a Lofargram: (using a second Hough transform).

3. Normalization and Denormalization

In the Hough transform negative values of ρ might be generated. This will cause errors in finding clusters when transfering these ρ values into binary format to be displayed. Hence the ρ negative values must be normalized to positive values. The computation was mentioned in chapter II. The positions of the clusters obtained from the LAS routine provide the ρ values and θ values, respectively. Since they will not be the original values, inverse processing (denormalization) in accordance with the equation (3.5) and equation (3.6) is required to recover the original ρ's and θ's.

4. Quantization

The accumulation array is quantized along the ρ and θ axes. For instance, assuming that the image space is a 256×256 array, then the accumulative array should be of the same dimension. Basically, the larger the size of the parameter accumulation array, the more accurate the procedure will be.

5. Experience in Using LAS

In order to use LAS, the splitting technique is needed to split the input image into ρ band and θ band images. But, the LAS routines have difficulties in finding the correct positions of the clusters.

6. Experience in Using Sorting

Sorting technique is an efficient method in finding clusters by setting an appropriate threshold. But, the disadvantage is that nearby clusters can not be treated as one cluster. This is the inherent problem of the Sorting technique. To solve this problem, cluster analysis needs to be improved. Of course, a threshold is required to make the similarity decision. From experience, using a similarity measure of $0.7 \sim 0.9$ yields good result. The theoretical consideration for an optimal similarity measure is left as future work.

7. Noise Tolerance

The Hough transform is able to handle noisy signals. The signal at a SNR of -3dB was reconstructed.

V. CONCLUSION

The purpose of this research was to extend the Hough transform method to the detection of spectral tracks of underwater objects. Data used were artificial images generated by computer programs and Lofargrams. Subjects in this study included the image space transformation, clustering methods, and reconstruction.

A modified Hough transform method was presented to yield better results in reconstruction by applying segmentation to decrease the noise effects. The clustering was an important analysis technique to search for significant distributions of the the cumulative array. This step significantly influences the reconstruction. Though applying the LAS failed to find the accurate cluster sites, an alternative way of using Sorting can make it possible.

The track detection of a moving object is an interesting problem, and is of interest in military applications. This thesis discusses a practical procedure to achieve tracking analysis of moving objects. The original procedure was shown not to work well with real images. An improved algorithm yielded better results. The source code program is listed in Appendix C. The following conclusions can be drawn from this research.

- The Hough transform methodology can be used for tracking of dynamic lines. This procedure has some tolerance to noise.
- The LAS routines did not provide satisfactory results.
- The Sorting technique provides a way to find the sites of the clusters.
- The Sorting technique has difficulty in locating a single cluster point per track in the parameter space, and hence reconstruction results into many (false) lines.
- A threshold factor needs to be determined when using the Sorting technique.

- The output from the line tracker can be fed into further analysis systems.

APPENDIX A. NOISE AND SIGNAL GENERATION SOURCE CODE

This appendix contains the Fortran source codes which can be used in generating the artificial noise and signal image. Reader can create the desired image by changing the necessary coefficients.

```
        program sn-gen
        byte a(256)/256*0/,y(256)/256*0/,b(256)/256*0/,w(256)/256*0/
        integer m(10),d(10),f(256,256),u(10)
        integer h(256,256),max1,min1,ab,s,aa
        integer ipd1,summ,time,gray,sumf,dbvalue
        integer mix(256,256),r(256),q(256),g(256,256),
        isy,s,aa,c,ic,ipd,sums
        real ab1,aa1,fac.dbf,rmss1,adj(256,256),
        adjmix(256,256),y1(256)
        real rmss,rmsn,db,index,ipd2
        open(unit=1,name='sn256.dat',type='new',access
     1  =direct',recordsize=64)
        open(unit=2,name='n256.dat',type='new',access=
     1  direct',recordsize=64)
        open(unit=3,name='s256.dat',type='new',access=
     1  direct',recordsize=64)
        open(unit=4,name='mix256.dat',type='new',access=
     1  direct',recordsize=64)
        byte a(128)/128*0/,y(128)/128*0/,b(128)/128*0/,w(128)/128*0/
        integer m(10),d(10),f(128,128),u(10)
        integer ab,h(128,128),max1,min1
        real ab1,aa1,fac
        real dbf,rmss1,rmsn1,adj(128,128),adjmix(128,128),y1(128)
        integer ipd1,summ,time,gray,sumf,dbvalue
        integer mix(128,128),r(128),q(128)
        integer isy,g(128,128),s,aa
        real rmss,rmsn,db,index,ipd2
        integer c,ic,ipd,sums
c**********************************************************************
c**** creat signal ****************************************************
c************************
        dbvalue=3
        m(1)=0
        m(2)=0
        d(1)=30
        d(2)=-20
        time=40
        gray=100
        c=2
        sums=0
        summ=0
        aa=0
        ic=0
        s=11
        isy=128
        do 9 j=1,c
        do 10 i=1,isy
              f(i,j)=(m(j)*(i-64)+d(j))+64
10       continue
9        continue
        print 100,f(1,1),f(1,2),f(2,1),f(2,2)
100      format(1x,3(i5,1x),i5)
        do 11 j=1,isy
              do 12 l=1,c
              u(l)=f(j,l)
                    do 14 k=1,isy
                          if(k.eq.u(l)) g(k,j)=gray
14                     continue
12             continue
11       continue
        do 15 j=1,isy
              do 16 i=1,isy
                    ipd=g(i,j)
                    if(ipd.eq.gray) then
                    ic=ic+1
                    sums=ipd**2+sums
                    endif
16             continue
15       continue
        rmss=sums**(0.5)
        print 17,ic,rmss
17       format(1x,'ic =',i5,3x,'rmss = ',f7.1)
        do 20 j=1,isy
              do 21 i=1,isy
              a(i)=g(i,j)
21             continue
              write(1'j) a
20       continue

c**********************************************************************
c****  creat noise ****************************
c******************************
        do 50 j=1,isy
              do 51 i=1,isy
                    y(i)=(ran(s)-0.5)*time
                    ab=y(i)
                    aa=ab**2+aa
                   h(i,j)=y(i)
51             continue
              write(2'j) y
50       continue
                    rmsn=aa**(0.5)
        print 52,rmsn
```

```
52     format(1x,'rmsn = ',f7.2)

c**********************************************************************
c**** add the signal with the noise **********
c*********************************************
      do 60 j=1,isy
            do 61 i=1,isy
                  mix(i,j)=g(i,j)+h(i,j)
                  ipd1=mix(i,j)
            if(mix(i,j).gt.max1) max1=mix(i,j)
            if(mix(i,j).lt.min1) min1=mix(i,j)
61           continue
60     continue
      fac=255/(max1-min1)
      do 62 j=1,isy
      do 63 i=1,isy
            mix(i,j)=(mix(i,j)-min1)*fac
                  if(mix(i,j).gt.127) then
                  mix(i,j)=mix(i,j)-255
                  else
                  mix(i,j)=mix(i,j)
                  endif
            b(i)=mix(i,j)
63     continue
            write(3'j) b
62     continue
      sm=summ**(0.5)
      db=20*LOG(sm/rmsn)
      index=rmss/(rmsn*(EXP(.1151293*dbvalue)-1))
      print 53,db,index,sm
53     format(1x,'db = ',f7.2,3x,'index = ',f7.2,1x,'sm=',f7.2)
c*******************************************************************
C**** redefine db to assigned DB ****************************
c*************** *********************************************
      sumf=0
      aa1=0.
      do 70 j=1,isy
            do 71 i=1,isy
                  adj(i,j)=index*h(i,j)
                  y1(i)=adj(i,j)
                  ab1=y1(i)
                  aa1=ab1**2+aa1
71           continue
70     continue
                  rmsn1=aa1**(0.5)
      print 200,rmsn1
200    format(1x,'rmsn1= ',f7.2)
      do 73 j=1,isy
            do 74 i=1,isy
                   adjmix(i,j)=g(i,j)+adj(i,j)
                  ipd2=adjmix(i,j)
                  sumf=ipd2**2+sumf
```

```
                    if(adjmix(i,j).gt.127) then
                    adjmix(i,j) = adjmix(i,j)-255
                    else
                    adjmix(i,j) = adjmix(i,j)
                    endif
                    w(i) = jnint(adjmix(i,j))
74              continue
      write(4'j) w
73     continue
      rmss1 = sum**(0.5)
      dbf= 20*LOG(rmss1/rmsn1)
      print 75,dbf,rmss1
75     format(1x,'dbf = ',f7.2,1x,'rmss1  = ',f7.2)
      end
```

APPENDIX B.   COMMAND OF ILS ARTIFICIAL IMAGE GENERATION

In this appendix, a series of commands will be presented as an example to generate any signal with appropriate random noise as follows.

- 1: SFIL 17
- 2: $CTX 128
- 3: STFU SWD 1,256,100;256,,,5
- 4: SINA SF256
- 5: SINA HMY
  :
- 6: $INA N128
- 7: $SDI LM1.5
- 8: SFIL 17
- 9: SFIL S19
- 10: SFIL SB20
- 11: $NSI N1,256,,8
- 12: $FIL 19
- 13: SFIL S21
- 14: $CTX 256
- 15: SOPN S
- 16: $SRE 1,128

- 17: $DRE S1,3
- 18: $FIL 21
- 19: $FIL S22
- 20: $OPN S
- 21: $FFT P1,128,,1
- 22: $FIL 22
- 23: DTO RFN;WFRONT.dat
- 24: $R SR2I

From 1 to 7, the signal will be generated. From 8 to 11, the noise will be generated. From 12 to 17, the mixed signal will be recorded. From 18 to 24, the FFT will be executed. The last step is to transfer ASCII code into binary code and normalize for the display.

APPENDIX C. MAIN SOURCE CODE PROGRAM

```
        program mod5
      byte a(256),p(256)
      integer zz(16),split/64/,split1/4/
      real threshold/.95/
      integer k,m,c,u(16),max,max1,ta(256),tc(256),x1(16,16)
      integer max2,min2,max3,isy1/256/
      integer ip(256,256),jrt(256,256),io,it,ir,ipd,td(256)
      real mean,mean1
      integer adx,ady,adt,adu,ip1(16,16),jrt1(16,16)
      real tb(256),x0,y0,row(256),x(16,16)
      real sc(16,2),dth,th,rowm,row0,drow,factor
      integer*4 isx/4/,isy/4/,ith/4/,iro/4/
      open(unit=1,name='beam00t.dat',status='old',access='direct'
     1 ,recordsize=64)
      open(unit=3,name='t1283.dat',status='new',access='direct'
     1 ,recordsize=64)
      open(unit=11,name='rcbeam00t.dat',status='new',access='direct'
     1 ,recordsize=64)
c      ****read in the image data & convert to integer.****************
       sum=0
       icc=0
       do 10 j=1,isy1
       read(1'j) a
         do 20 i=1,isy1
            ip(i,j)=a(i)
               jrt(i,j)=0
20      continue
10      continue
       do 21 j=1,isy1
       do 22 i=1,isy1
            if(ip(i,j).lt.0) ip(i,j)=ip(i,j)+256
22      continue
21      continue
       do 23 j=1,isy1
      do 24 i=1,isy1
               sum=sum+ip(i,j)
24      continue
23      continue
               mean=sum/(isy1*isy1)+20
c      print 801,mean
c801    format(1x,'mean=',f5.1)
       do 25 jj=1,split
               do 25 ii=1,split
                       sum1=0
               do 26 l=1,split1
                       do 26 k=1,split1
                       sum1=sum1+ip((ii-1)*split1+k,(jj-1)*split1+l)
```

```
                ip1(k,l)=ip((ii-1)*split1+k,(jj-1)*split1+l)
26          continue
      icc=icc+1
                mean1=sum1/(split1**2)
c     print 800,icc,mean1
c800    format(1x,'icc=',i6,2x,'mean1=',f6.1)
           if(mean1.ge.mean) then
c**********hough transform**********************
      call ht2(ip,jrt,sc,iro,isx,isy,ith,drow,row0,x0,y0,ip1,jrt1)
c****************************************************
      max=jrt1(1,1)
      min=jrt1(1,1)
      do 30 j=1,isy
      do 40 i=1,isy
      if(jrt1(i,j) .gt.max) max=jrt1(i,j)
      if(jrt1(i,j) .lt. min) min=jrt1(i,j)
      ta(i)=jrt1(i,j)
 40    continue
 30    continue
      pai=3.1415926
      c=0
      max1=jnint(threshold*max)
      do 90 j=1,isy
           do 100 i=1,isy
                if(jrt1(i,j) .gt. max1) then
                 c=c+1
           tb(c)=(i-1)*180/isy
                tc(c)=j
                endif
100         continue
90     continue
      do 110 j=1,c
            row(j)=tc(j)*drow-row0
110    continue
      do 120 j=1,c
      do 130 i=1,isy
      x(i,j)=x0+(row(j)-(y0-i)*sin(tb(j)*pai/180)/cos(tb(j)*pai/180))
      x1(i,j)=jnint(x(i,j))
      td(i)=x1(i,j)
130    continue
120    continue
      do 140 j=1,isy
           do 150 l=1,c
                u(l)=x1(j,l)
150         continue
           do 160 m=1,c
                do 161 k=1,isy
                if(k.eq.u(m)) x1(k,j)=255
161               continue
160         continue
140    continue
      do 162 j=1,isy
      do 163 i=1,isy
            if(x1(i,j).ne.255) then
                  x1(i,j)=0
            else
                  x1(i,j)=x1(i,j)
            endif
      jrt((ii-1)*split1+i,(jj-1)*split1+j)=x1(i,j)
                  zz(i)=jrt((ii-1)*split1+i,(jj-1)*split1+j)
163    continue
162    continue
      endif
25     continue
      do 70 j=1,isy1
            do 80 i=1,isy1
                  if(jrt(i,j) .ge. 127) then
                  jrt(i,j)=jrt(i,j)-256
                  else
                  jrt(i,j)=jrt(i,j)
                   endif
                  a(i)=jrt(i,j)
80           continue
                  write(11'j) a
70     continue
c*************************************************************
      call mod1
c*************************************************************
      call t2563
c************************************************************
      call xx
c***************************************************************************
      call clustering
c***************************************************************************
      end
c**************** *********************** *************************
c***************************************************************************
      subroutine mod1
      byte a(256),p(256)
      integer zz(16),split/128/,split1/2/
      real threshold/.95/
      integer k,m,c,u(16),max,max1,ta(256),tc(256),x1(16,16)
      integer max2,min2,max3,isy1/256/
      integer ip(256,256),jrt(256,256),io,it,ir,ipd,td(256)
      real mean,mean1
      integer adx,ady,adt,adu,ip1(16,16),jrt1(16,16)
      real tb(256),x0,y0,row(256),x(16,16)
      real sc(16,2),dth,th,rowm,row0,drow,factor
      integer*4 isx/2/,isy/2/,ith/2/,iro/2/
      open(unit=11,name='rcbeam00t.dat',status='old',access='direct'
    1 ,recordsize=64)
      open(unit=12,name='rcbm1.dat',status='new',access='direct'
    1 ,recordsize=64)
c     ****read in the image data & convert to integer.****************
```

```
      sum=0
      icc=0
      do 10 j=1,isy1
      read(1'j) a
        do 20 i=1,isy1
           ip(i,j)=a(i)
              jrt(i,j)=0
20      continue
10      continue
       do 21 j=1,isy1
       do 22 i=1,isy1
              if(ip(i,j).lt.0) ip(i,j)=ip(i,j)+256
22      continue
21      continue
       do 23 j=1,isy1
       do 24 i=1,isy1
              sum=sum+ip(i,j)
24      continue
23      continue
              mean=sum/(isy1*isy1)+25
       do 25 jj=1,split
              do 25 ii=1,split
                     sum1=0
              do 26 l=1,split1
                     do 26 k=1,split1
                     sum1=sum1+ip((ii-1)*split1+k,(jj-1)*split1+l)
                     ip1(k,l)=ip((ii-1)*split1+k,(jj-1)*split1+l)
26             continue
        icc=icc+1
                     mean1=sum1/(split1**2)
               if(mean1.ge.mean) then
c***********hough transform**************************
        call ht2(ip,jrt,sc,iro,isx,isy,ith,drow,row0,x0,y0,ip1,jrt1)
c*****************************************************************
        max=jrt1(1,1)
        min=jrt1(1,1)
        do 30 j=1,isy
        do 40 i=1,isy
        if(jrt1(i,j) .gt.max) max=jrt1(i,j)
        if(jrt1(i,j) .lt. min) min=jrt1(i,j)
        ta(i)=jrt1(i,j)
  40     continue
  30     continue
        pai=3.1415926
        c=0
        max1=jnint(threshold*max)
        do 90 j=1,isy
               do 100 i=1,isy
                      if(jrt1(i,j) .gt. max1) then
                       c=c+1
               tb(c)=(i-1)*180/isy
                      tc(c)=j
```

```
                  endif
100          continue
90      continue
        do 110 j=1,c
             row(j)=tc(j)*drow-row0
110      continue
        do 120 j=1,c

        do 130 i=1,isy
        x(i,j)=x0+(row(j)-(y0-i)*sin(tb(j)*pai/180)/cos(tb(j)*pai/180))
        x1(i,j)=jnint(x(i,j))
        td(i)=x1(i,j)
130      continue
120      continue
        do 140 j=1,isy
             do 150 l=1,c
                  u(l)=x1(j,l)
150          continue
             do 160 m=1,c
                  do 161 k=1,isy
                  if(k.eq.u(m)) x1(k,j)=255
161                continue
160          continue
140      continue
        do 162 j=1,isy
        do 163 i=1,isy
             if(x1(i,j).ne.255) then
                  x1(i,j)=0
             else
                  x1(i,j)=x1(i,j)
             endif
        jrt((ii-1)*split1+i,(jj-1)*split1+j)=x1(i,j)
                  zz(i)=jrt((ii-1)*split1+i,(jj-1)*split1+j)
163      continue
162      continue
         endif
25       continue
         do 70 j=1,isy1
              do 80 i=1,isy1
                   if(jrt(i,j) .ge. 127) then
                   jrt(i,j)=jrt(i,j)-256
                   else
                   jrt(i,j)=jrt(i,j)
                    endif
                   a(i)=jrt(i,j)
80            continue
                   write(12'j) a
70       continue
         end
c*********************************************************************
c*********** for double call *************************************
c*********************************************************************
```

```
      subroutine ht2(ip,jrt,sc,isx,isy,ith,iro,drow,row0,x0,y0,ip1,jrt1)
c*************************************************************************
      dimension ip1(isx,isy)
      dimension jrt1(ith,iro)
      real sc(ith,2)
      real x0,y0
      real pai
      pai=3.1415926
      jsx=isx
      jsy=isy
      jth=ith
      jro=iro
      do 100 j=1,jro
      do 100 i=1,jth
        jrt1(i,j)=0
100     continue
      dth=pai/float(jth)
      do 10 i=1,jth
        th=float(i-1)*dth
        sc(i,1)=cos(th)
        sc(i,2)=sin(th)
10     continue
      rowm=sqrt(float(jsx*jsx)+float(jsy*jsy))
      row0=rowm/2.0
      drow=rowm/float(jro)
      x0=float((jsx+2)/2)
      y0=float((jsy+2)/2)
      do 40 iy=1,jsy
          y=iy
        do 30 ix=1,jsx
            ipd=ip1(ix,iy)
            if(ipd.le.0) go to 30
            x=ix
            do 20 it=1,jth
                row=(x-x0)*sc(it,1)+(y0-y)*sc(it,2)
                r=(row+row0)/drow
                ir=jnint(r)
                jrt1(it,ir)=jrt1(it,ir)+ipd
20             continue
30         continue
40     continue
      return
      end
c****************************************************************************
c*************************************************************************
      subroutine t2563
      byte a(256),p(256)
      integer k,m,c,u(256),max,max1,ta(256),tc(256),x1(256,256)
      integer max2,min2,max3
      integer ip(256,256),jrt(256,256),io,it,ir,ipd,td(256)
      real tb(256),x0,y0,row(256),x(256,256)
      real sc(256,2),dth,th,rowm,row0,drow,factor,threshold
      integer*4 isx/256/,isy/256/,ith/256/,iro/256/
      open(unit=12,name='rcbml.dat',status='old',access='direct'
     1 ,recordsize=64)
      open(unit=13,name='rcbmlast.dat',status='new',access='direct'
     1 ,recordsize=64)
c     ****read in the image data & convert to integer.*****************
      threshold=0.32
      do 10 j=1,isy
          read(12'j) a
        do 20 i=1,isy
            ip(i,j)=a(i)
20        continue
10     continue
      do 21 j=1,isy
        do 22 i=1,isy
            if(ip(i,j).lt.0) ip(i,j)=ip(i,j)+256
22       continue
21     continue
c***********hough transform**************************
      call ht1(ip,jrt,sc,iro,isx,isy,ith,drow,row0,x0,y0)
c****************************************************
      max=jrt(1,1)
      min=jrt(1,1)
      do 30 j=1,isy
        do 40 i=1,isy
            if(jrt(i,j) .gt.max) max=jrt(i,j)
            if(jrt(i,j) .lt. min) min=jrt(i,j)
            ta(i)=jrt(i,j)
40       continue
30     continue
      pai=3.1415926
      c=0
      max1=jnint(threshold*max)
      print 81,max
81     format(1x,'max = ',i5)
      print 82,max1
82     format(1x,'max1 = ',i5)
      do 90 j=1,isy
        do 100 i=1,isy
            if(jrt(i,j) .gt. max1) then
            c=c+1
            tb(c)=(i-1)*180/isy
            tc(c)=j
            endif
100       continue
90     continue
      print 91,c
91     format(1x,'counter = ',i5)
      do 110 j=1,c
            row(j)=tc(j)*drow-row0
110    continue
      print 92,row(1)
```

```
92     format(1x,'row1=',f6.1)
      print 93,row(2)
93     format(1x,'row2=',f6.1)
      do 120 j=1,c
        do 130 i=1,isy
       x(i,j)=x0+(row(j)-(y0-i)*sin(tb(j)*pai/180)/cos(tb(j)*pai/180))
        x1(i,j)=jnint(x(i,j))
        td(i)=x1(i,j)
130    continue
120    continue
      do 140 j=1,isy
            do 150 l=1,c
            u(l)=x1(j,l)
150           continue
            do 160 m=1,c
              do 161 k=1,isy
                 if(k.eq.u(m)) x1(k,j)=255
161             continue
160           continue
140    continue
      print 151,u(1),u(2)
151    format(1x,i5,1x,i5)
      do 162 j=1,isy
        do 163 i=1,isy
            if(x1(i,j).ne.255) x1(i,j)=0
163      continue
162    continue
      do 70 j=1,isy
      do 80 i=1,isy
            if(x1(i,j) .ge. 127) then
            x1(i,j)=x1(i,j)-256
            else
            x1(i,j)=x1(i,j)
            endif
            p(i)=x1(i,j)
80     continue
      write(13'j) p
70     continue
      return
      end
c*********************************************************************
      subroutine ht1(ip,jrt,sc,isx,isy,ith,iro,drow,row0,x0,y0)
c*********************************************************************
      dimension ip(isx,isy)
      dimension jrt(ith,iro)
      real sc(ith,2)
      real x0,y0
      real pai
      pai=3.1415926
      jsx=isx
      jsy=isy
      jth=ith
      jro=iro
      do 100 j=1,jro
      do 100 i=1,jth
        jrt(i,j)=0
100    continue
      dth=pai/float(jth)
      do 10 i=1,jth
        th=float(i-1)*dth
        sc(i,1)=cos(th)
        sc(i,2)=sin(th)
10     continue
      rowm=sqrt(float(jsx*jsx)+float(jsy*jsy))
      row0=rowm/2.0
      drow=rowm/float(jro)
      x0=float((jsx+2)/2)
      y0=float((jsy+2)/2)
      do 40 iy=1,jsy
          y=iy
        do 30 ix=1,jsx
            ipd=ip(ix,iy)
            if(ipd.le.0) go to 30
            x=ix
          do 20 it=1,jth
              row=(x-x0)*sc(it,1)+(y0-y)*sc(it,2)
              r=(row+row0)/drow
              ir=jnint(r)
              jrt(it,ir)=jrt(it,ir)+ipd
20          continue
30     continue
 40    continue
      return
      end
c*****************************************************************************
c*****************************************************************************
      subroutine xx
      byte a(256)
      real iiR(256,4096)
      integer iiy,k,m,ic
      integer ip(256,256),jrt(256,256),io,it,ir,ipd,ia(256)
      real ta(256),tb(256),max,min
      real sc(2,256),dth,th,rowm,row0,drow,ib(256),factor,fa1,fa2
      integer*4 isx/256/,isy/256/,ith/256/,iro/256/
      open(unit=13,name='rcbmlast.dat',status='old',access='direct'
     1 ,recordsize=64)
      open(unit=14,name='rcbmclu.dat',status='new',access='direct'
     1 ,recordsize=64)
c      ****read in the image data & convert to integer.*****************
      do 10 j=1,256
      read(3'j) a
       do 20 i=1,256
         ip(i,j)=a(i)
           if(ip(i,j).lt.0) then
```

```
                ip(i,j)=ip(i,j)+256
                endif
 20      continue
 10      continue
        max1=0
        min1=0
        do 21 j=1,256
        do 22 i=1,256
        if(ip(i,j) .gt.max1) max1=ip(i,j)
        if(ip(i,j) .lt. min1) min1=ip(i,j)
 22      continue
 21      continue
        print 900,max1,min1
900      format(1x,2i7)
c***********hough transform*************************
        call ht2(ip,jrt,sc,isx,isy,ith,iro,ic)
c*****************************************************
        max=jrt(1,1)
        min=jrt(1,1)
        do 30 j=1,256
        do 40 i=1,256
        if(jrt(i,j) .gt.max) max=jrt(i,j)
        if(jrt(i,j) .lt. min) min=jrt(i,j)
 40      continue
 30      continue
                factor=255.0/(max-min)
        do 50 j=1,256
        do 60 i=1,256
                jrt(i,j)=jnint((jrt(i,j)-min)*factor)
 60      continue
 50      continue
        threshold=255*0.2
        do 81 j=1,256
                do 82 i=1,256
                if(jrt(i,j).lt.threshold) jrt(i,j)=0
 82      continue
 81      continue
        do 70 j=1,256
        do 80 i=1,256
                if(jrt(i,j) .ge. 127) then
                a(i)=jrt(i,j)-255
                else
                a(i)=jrt(i,j)
                endif
 80       continue
        write(4'j) a
 70      continue
        end
c*******************************************************************
c*******************************************************************
        subroutine clustering
        byte a(256),p(256)
```

```
      integer zz(8),split/128/,split1/2/,w,z,t(256)
      real threshold
      integer k,m,c,u(4),max,max1,ta(256),tc(256),x1(4,4)
      integer max2,min2,max3,isy1/256/,ct
      integer ip(256,256),jrt(256,256),io,it,ir,ipd,td(128)
      real mean,mean1,s(256,256),te1
      integer ite,tf,ib,aa(256,256),cou
      integer adx,ady,adt,adu,ip1(4,4),jrt1(4,4)
      real tb(256),row(128),x(4,4)
      real sc(4,2),dth,th,rowm,row0,drow,factor
      integer*4 isx/2/,isy/2/,ith/2/,iro/2/
       open(unit=14,name='rcbmclu.dat',status='old',access='direct'
     1 ,recordsize=64)
      open(unit=15,name='clubm.dat',status='new',access='direct'
     1 ,recordsize=64)
c      ****read in the image data & convert to integer.*****************
      cou=0
      do 10 j=1,isy1
      read(4'j) a
        do 20 i=1,isy1
           ip(i,j)=a(i)
              if(ip(i,j).lt.0) then
                     ip(i,j)=ip(i,j)+255
              endif
              jrt(i,j)=0
              t(i)=ip(i,j)
              if(t(i).gt.0) cou=cou+1
 20    continue
 10    continue
      print 902,cou
902    format(1x,i6)
      do 800 w=1,10
      max=0
      min=0
      do 23 j=1,isy1
      do 24 i=1,isy1
              if(ip(i,j).gt.max) max=ip(i,j)
              if(ip(i,j).lt.min) min=ip(i,j)
24     continue
23     continue
      print 901,max,min
901    format(1x,'max=',i6,'min=',i6)
c*******************************************
      if(max.gt.min) then
      ct=0
      do 100 j=1,isy1
              do 101 i=1,isy1
              if(ip(i,j).ge.max) then
                     ct=ct+1
c                    te1=(i-1)*2*180/isy1
                     ite=i
                     tf=j
```

```
                    ib=ite**2+tf**2
                endif

101                 continue
100         continue
            ite=ite
            ib=ib
            do 102 j=1,isy1
                do 103 i=1,isy1
                    if(ip(i,j).gt.0) then
                        c=i**2+j**2
                            aa(i,j)=(ite-i)**2+(tf-j)**2
                        s(i,j)=(b+c-aa(i,j))/(b+c+aa(i,j))
                            if(s(i,j).lt.0.89) then
                            ip(i,j)=ip(i,j)
                            else
                            ip(i,j)=0
                            endif
                    endif
                    ip(i,j)=ip(i,j)
103                     continue
102                     continue
            jrt(ite,tf)=255
            endif

800         continue
            do 106 j=1,isy1
                do 107 i=1,isy1
                    if(jrt(i,j).gt.127) then
                    jrt(i,j)=jrt(i,j)-256
                    else
                    jrt(i,j)=jrt(i,j)
                    endif
                    a(i)=jrt(i,j)
107                 continue
            write(5'j) a
106         continue
            call reconst
            end
c**************************************************
c************************************************************
            subroutine reconst
            byte a(256),p(256)
            integer k,m,c,u(256),max,max1,ta(256),tc(256),x1(256,256)
            integer x0/129/,y0/129/,tb(256)
            integer ip(256,256),jrt(256,256),io,it,ir,ipd,td(256)
            real row(256),x(256,256)
            real sc(256,2),dth,th,rowm,row0,drow,factor,threshold
            integer*4 isx/256/,isy/256/,ith/256/,iro/256/
            open(unit=15,name='clubm.dat',status='old',access='direct'
         1  ,recordsize=64)
             open(unit=16,name='rcclubm.dat',status='new',access='direct'
     1  ,recordsize=64)
c       ****read in the image data & convert to integer.*****************
        drow=2**(0.5)
        row0=128*drow
        do 10 j=1,isy
        read(5'j) a
          do 20 i=1,isy
             jrt(i,j)=a(i)
 20      continue
 10      continue
        do 21 j=1,isy
        do 22 i=1,isy
               if(jrt(i,j).lt.0) jrt(i,j)=jrt(i,j)+256
22       continue
21       continue
c*************  reconstruction  ************************
c*****************************************************
        pai=3.1415926
        c=0

        do 90 j=1,isy

        do 100 i=1,isy
               if(jrt(i,j) .gt. 0) then
                c=c+1
               tb(c)=i
               tc(c)=j
               endif
100      continue
90       continue
        print 91,c
91       format(1x,'counter = ',i5)
        do 110 j=1,c
               row(j)=tc(j)*drow-row0
110      continue
        print 92,row(1)
92       format(1x,'row1=',f6.1)
        print 93,row(2)
93       format(1x,'row2=',f6.1)
        do 120 j=1,c

        do 130 i=1,isy
        x(i,j)=x0+(row(j)-(y0-i)*sin(tb(j)*pai/256))/cos(tb(j)*pai/256)
        x1(i,j)=jnint(x(i,j))
        td(i)=x1(i,j)
130      continue
120      continue
        do 140 j=1,isy
               do 150 l=1,c
               u(l)=x1(j,l)
150              continue
```

```
              do 160 m=1,c
              do 161 k=1,isy
                  if(k.eq.u(m)) x1(k,j)=255
161            continue
160            continue
140      continue
         print 151,u(1),u(2)
151      format(1x,i5,1x,i5)
         do 162 j=1,isy
         do 163 i=1,isy
              if(x1(i,j).ne.255) x1(i,j)=0
163      continue
162      continue
         do 70 j=1,isy
         do 80 i=1,isy
              if(x1(i,j) .ge. 127) then
              x1(i,j)=x1(i,j)-256
              else
              x1(i,j)=x1(i,j)
              endif
              p(i)=x1(i,j)
80       continue
         write(6'j) p
70       continue
         return
         end
```

LIST OF REFERENCES

1. *IM-4000 Image Manager Manual,* METSAT, Inc., 515 South Howes, Fort Collins, CO 80521, 1988.

2. *LAS User's Guide*, Version 4.0, Godadard Space 'Flight Center, Greenbelt, Maryland, September, 1987.

3. Duda, R. O., and Hart, P. E., "Use of Hough Transformation to Detect Lines and Curves in Pictures," *Communication of the ACM*, Vol. 15, No. 1, pp. 11-15, January, 1972.

4. Deans, S. R., "Hough Transform from the Radon Transform," *IEEE Trans. on Pattern Analysis and Machine Inteligence*, Vol. PAMI-3, No. 2, pp. 185-188, March, 1981.

5. Jain, A. K., *Fundamentals of Digital Image Processing*, Prentice Hall, N.J., 1989.

6. Ballard, D. H., and Brown, C. M., *Computer Vision*, Prentice Hall, N.J., 1982.

7. *ILS User's Guide*, Signal Technology, Inc., 5951 Encina Rd, Goleta, CA 93117, 1987.

8. Lamdan, Y., Schwartz J. T., & Wolfson, H. I., "Object Recognition by Affine Invariant Matching," Present at *IEEE Conference on Computer Vision and Pattern Recognition*, June, 1988.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Department Chairman, EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Professor Chin-Hwa Lee, EC/Le Naval Postgraduate School Monterey, CA 93943-5000	2
5.	Professor Ralph Hippenstiel, EC/Hi Naval Postgraduate School Monterey, CA 93943-5000	2
6.	Kao Chang-Lung 7,Lane 72, Tzu-Lin, Chin-Tan, 23180, Sin-Tien, Taipei, Taiwan, R.O.C.	1
7.	Wang Chen-Shan 4-1,Lane 45, Alley 217, An-Lei Rd, Chung-Ho, 23538, Taipei, Taiwan, R.O.C.	3
8.	Dan Zulaica, EC Naval Postgrdurate School Monterey, CA 93943-5000	1
9.	Chen Chih-Lyeu 4f, 10, Lane 70, Yu-Feng st, Ku-Hsan Kaohsiung, Taiwan, R.O.C.	1